

FUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SC
MONTEREY, CALIF. 93940



NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

A HIGH RESOLUTION
AMMUNITION RESUPPLY MODEL

by

Peter J. Bucha

and

Thomas J. McGrann

March 1982

Thesis Co-Advisors:

S.H. Parry

J.K. Hartman

Approved for public release: distribution unlimited

T204547

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A HIGH RESOLUTION AMMUNITION RESUPPLY MODEL		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis March 1982
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Peter J. Bucha Thomas J. McGrann		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE March 1982
		13. NUMBER OF PAGES 268
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release: distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Simulation Logistics Simulation Combat Model Ammunition Simulation STAR Model		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This thesis presents a computer simulated model of ammunition resupply in a U.S. combat battalion. The model is based on current ammunition resupply doctrine and has been designed as a stand-alone simulation. Additionally, this model has been structured to parrallel the Simulation of Tactical Alternative Responses (STAR) model so that future enhancements might include its full integration into the STAR model. When an integration		

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

#20 - ABSTRACT - CONTINUED

is accomplished, the important dimension of combat service support will become an influencing factor in the decision making process at all levels of the combat model.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

Approved for public release: distribution unlimited

A High Resolution Ammunition Resupply Model

by

Peter J. Bucha
Captain, United States Army
B.S., United States Military Academy, 1972
and

Thomas J. McGrann
Captain, United States Army
B.S., United States Military Academy, 1972

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the
NAVAL POSTGRADUATE SCHOOL
March, 1982

ABSTRACT

This thesis presents a computer simulated model of ammunition resupply in a U.S. combat battalion. The model is based on current ammunition resupply doctrine and has been designed as a stand-alone simulation. Additionally, this model has been structured to parallel the Simulation of Tactical Alternative Responses (STAR) model so that future enhancements might include its full integration into the STAR model. When such an integration is accomplished, the important dimension of combat service support will become an influencing factor in the decision making process at all levels of the combat model.

TABLE OF CONTENTS

I.	INTRODUCTION	14
A.	GENERAL	14
B.	LOGISTICS MODELS	16
C.	THESIS OBJECTIVES	17
D.	THESIS ORGANIZATION	20
II.	STATE OF THE ART	22
A.	INTRODUCTION	22
B.	AMMUNITION SUPPORT IN TODAY'S ARMY	22
C.	EXISTING ARMY MODELS	25
1.	HELAPS II	25
a.	Purpose	26
b.	Characteristics	27
c.	Assumptions	28
d.	Input	29
e.	Output	30
f.	Strengths	30
g.	Weaknesses	32
2.	ARM	32
a.	Purpose	33
b.	Characteristics	33
c.	Assumptions	34
d.	Input	35

e.	Output	35
f.	Strengths	36
g.	Weaknesses	37
D.	LOGISTICS MODEL DEVELOPMENT AT NPS	37
1.	"Simulation and Analysis of Transport in Support of a Combat Unit" by John R. Kelley (1978) [Ref. 3]	38
a.	Assumptions	39
b.	Method of Evaluation	40
c.	Strengths	42
d.	Weaknesses	42
e.	Utility	43
2.	"Simulation of Tactical Alternative Responses" by W. S. Wallace and E. G. Hagewood (1978) [Ref. 4]	44
a.	Methodology	45
b.	Strengths	48
c.	Weaknesses	49
d.	Utility	49
3.	"A Dynamic Ammunition and Resupply Model in Support of the STAR Model" by Bruce G. Ripley (1979) [Ref. 5]	50
a.	Assumptions	51

b.	Characteristics	52
c.	Strengths	55
d.	Weaknesses	56
e.	Utility	57
4.	"A High Resolution Integrated Combat and Logistics Model" by D.G. Kirby and D.P. Schultz (1980) [Ref. 6]	58
a.	Assumptions	58
b.	Methodology	59
c.	Strengths	61
d.	Weaknesses	61
e.	Utility	62
III.	MODEL DESCRIPTION	63
A.	INTRODUCTION	63
B.	THE BATTLE	64
C.	REQUESTS FOR RESUPPLY	67
1.	Level of Need (LON)	68
2.	Requests for Resupply	71
3.	Imperfect Logistics Information	72
4.	Assumptions	74
D.	RESUPPLY	75
1.	Battalion Distribution	75
2.	Company Distribution	76

3. Platoon Distribution	77
4. Assumptions	77
E. REDISTRIBUTION	79
1. Redistribution Due To Relocation	79
2. Redistribution Due To A Firepower Kill	79
3. Assumptions	80
IV. CONCLUDING REMARKS AND FUTURE ENHANCEMENTS	81
A. GENERAL	81
B. MODEL DEFICIENCIES	82
C. PATH OF FUTURE DEVELOPMENT	84
D. INTEGRATION INTO A COMBAT MODEL	86
APPENDIX A: DETAILED METHODOLOGY OF THE MODEL	89
A. INTRODUCTION	89
B. USE OF SIMSCRIPT II.5 IN THE MODEL	89
C. GENERAL MODEL METHODOLOGY	91
D. INPUT REQUIREMENTS AND THE INITIALIZATION OF DATA ARRAYS	92
1. Generating Resupply Requirements - The Battle	93
a. Ammunition Expenditures	94
b. Vehicle/Weapon Damage and Destruction	96
c. Unit Moves	97
2. Determining the Level of Need (LON)	97
a. Imperfect Information	98

b.	Initial Assets and Capacities	99
c.	Weapon LON's	100
d.	Platoon LON	102
e.	Company LON	104
3.	Requests for Resupply	108
a.	Weapon Systems	108
b.	Platoon	109
c.	Company	109
d.	Battalion	110
4.	Supply Response - Action by the S-4	111
a.	Availability of Supply and Transportation Assets	111
b.	Maximization of Shipping Space	112
c.	Adjustments Due to Priority Requisitions.	112
E.	RESUPPLY ACTIVITIES - RECEIPT OF SUPPLIES	113
APPENDIX B: PROGRAM DOCUMENTATION		115
A.	"PREAMBLE"	115
1.	Routines	116
2.	Events	116
3.	Entities	117
4.	Attributes	120
5.	Sets	132
6.	Global Variables	133

7.	Listing	138
B.	"MAIN"	140
C.	"ROUTINE BLU.CREATE"	141
D.	"ROUTINE PARAMETERS"	148
E.	"ROUTINE BASIC.LOAD"	152
F.	"ROUTINE W.AMMO"	159
G.	"ROUTINE P.CLASS.V"	165
H.	"ROUTINE COM.AMMO"	172
I.	"ROUTINE BATTLE"	180
J.	"ROUTINE UP.DATE"	185
K.	"ROUTINE FILE.UP.DATE"	189
L.	"ROUTINE LOAD.THE.TRUCKS"	193
M.	"ROUTINE WT.AND.CU"	200
N.	"ROUTINE PRI.RESUPPLY"	203
O.	"EVENT BAT.L.TIME"	209
P.	"EVENT MOVE"	211
Q.	"EVENT CO.RESUPPLY.ARR"	212
R.	"EVENT REDISTRIBUTE"	221
S.	"EVENT BN.ARRIVE"	226
T.	"EVENT UP.W.AMMO"	229
U.	"EVENT UP.PLT.AMMO"	231
V.	"EVENT UP.COM.AMMO"	234
W.	"EVENT B.UP.DATE"	236

X.	"EVENT STOP.SIMULATION"	237
Y.	"EVENT UP.S4.AMMO"	237
Z.	"EVENT FIREKILL"	256
LIST OF REFERENCES			263
BIBLIOGRAPHY			264
INITIAL DISTRIBUTION LIST			265

LIST OF FIGURES

1.	Ammunition Support Structure	24
2.	Resupply Process	92
3.	Weapon LON Example	102
4.	Platoon LON Example	105
5.	Company LON Example	107

ACKNOWLEDGMENT

We gratefully acknowledge the assistance of CPT Howard J. Carpenter, USA, Professor Hartman, and Professor Parry in the preparation of this thesis. Without their untiring efforts the research would not have progressed as far, nor would the end product have been as satisfying.

I. INTRODUCTION

A. GENERAL

Combat is a complex, multi-dimensional process, the nature of which defies simple description. In its broadest sense, this process is divided into land, sea, and air categories. The Army, for its part, focusing its attention on the air/land battle, subdivides the process further into Combat, Combat Support, and Combat Service Support functions. While it is generally recognized that a total picture of combat cannot be gained by looking at any single subdivision of the process, this segmentation overlays a conceptual framework that brings with it a degree of clarity necessary for analysis purposes.

Computer simulations modelling the air/land battle have naturally evolved along these segmented paths. They have, however, become so functionalized that they exist today as parallel and completely separate worlds seeking to model the same phenomenon. The degree to which this segmentation has developed was a natural consequence both of hardware limitations imposed by older generation computers, and by the very complexity of the combat process itself.

Modellers, in attempting to present a "total" picture of combat, have erected bridges between these paths with varying degrees of success. These bridges generally take the form of a Lanchester equation. In practice, a modeller concentrates on one path seeking resolution while portraying necessary parallel processes in this simplified mathematical form. This technique permits the modeller to capture at least some measure of the interaction of these paths on battle outcome, without necessitating the use of an unrealistic amount of computer time.

The direct consequence of this situation is that while usually answering questions directed at some particular segment, and amply answering the questions of who, what, and when at the interface points between paths, such models do not achieve the sharpened focus needed to answer the where, how, and why of the total process. Perhaps this last set of questions is avoided because the answer to them requires modelling command logic and synergistic interplay difficult to capture and quite conceivably beyond the scope of any one model's charter. At any rate, the loss of this detail prevents one from really answering basic questions as to the effectiveness and workability of the combined "total"

process. In essence, the fundamental question, "Can it be done?", is never adequately answered.

B. LOGISTICS MODELS

Current logistics models then, although running the gamut from high level, high resolution simulations to low level, low resolution simulations, must be characterized, in accordance with the above arguments, as single dimensional models. For, operating independently of any combat model, logistics models must generate the demands they respond to in some "artificial" way either externally or internally. Externally, data generated from a combat model is converted to some record of expenditures and front-ended into the model as a sequence of events. Internally, the combat process is expressed by some form of Lanchester equation which dynamically generates specific requirements within the context of the model itself. Output from these models is then designed to provide a record of activities completed over time as a measurement of an organization's ability to respond to a varied requirements load. Logistics effectiveness is then examined in aggregated terms, such as tons delivered per day, or miles travelled per day. Details of interplay between the combat and logistical processes are generally not available.

With the development of the Simulation of Tactical Alternative Responses (STAR) model at the Naval Postgraduate School (NPS), logisticians have been given a unique opportunity to investigate this dynamic interaction between combat and logistical processes at the most critical juncture, the battlefield. Since its initial coding in 1978, the model has been continually expanded to include an ever widening horizon of combat functions. Several preliminary attempts have been made to integrate logistics into the model, but as yet, little logistics is played.

C. THESIS OBJECTIVES

In light of the above considerations, a goal of developing a rudimentary high resolution logistics module was established, with the ultimate aim of the project being its eventual integration into the STAR model. After some consideration, the project's goal was redefined further and effort was centered on the development of an ammunition (Class V) model within a combat battalion. Ammunition was chosen both because of its critical impact on the battlefield and because the logic developed for this class of supply could easily be expanded to other classes. Such an attempt would represent at least a first step toward

capturing the subtle interaction between the combat and logistical processes. The project's planned approach was to have included:

- A review of previous work done at NPS in the area of resupply.
- A review of existing ammunition resupply models currently in use within the Army community.
- Familiarization with the workings of the STAR model with the objective of identifying the logical interface points for a logistics module.
- The design of an ammunition resupply model written in SIMSCRIPT II.5 that simulates the resupply process within a combat battalion.
- Integration of this logistics model into STAR at the appropriate points so that STAR generated data such as ammunition expended, vehicles killed, et cetera, would provide a driver for the logistics module.
- Incorporation of the information provided by the logistics module into the company and battalion commander decision logic of the STAR model.

Preliminary forays into the organization, use, and operational vicissitudes of the STAR model, however,

immediately identified and underscored the primary disjuncture between combat and service support models to be pace of battle. The plain fact is that the critical time window for analysis of operational effectiveness is different for the two types of models. High resolution combat models generally focus on the inner workings of a battle that may last less than four hours. Logistics models, in contrast, operate over time spans ranging from three to thirty days. In the current STAR model, battles generally terminate after three hours, well before the logistical network is even alerted for action. Planned expansion of the STAR model beyond the brigade battle now fought, however, makes the eventual inclusion of logistical factors both possible and necessary.

With these considerations in mind, the initial objectives of interfacing directly with the STAR model was modified and the planned logistics module was expanded to a full stand-alone, high resolution model. Supplementary objectives were established in order to achieve the completeness of a stand-alone model while keeping the ultimate objective of eventual interface with STAR in sight. These additional objectives were:

- Development of a detailed model that responds to requests for ammunition resupply, maintains a simplified stockage system, and models the movement of rounds down to the individual weapon.
- Maintenance of an appropriate level of resolution within the model. Eventual integration into the STAR model dictates that the resupply process must begin with and be based on knowledge of expenditures of a variety of ammunitions fired by any number of individual systems.
- Maintenance of a high degree of flexibility within the model. The model designed must be flexible enough to adapt to the wide variety of tables of organization and equipment that might be played in STAR.

D. THESIS ORGANIZATION

Chapter II provides an overview of the background research which went into the formation of this model. It includes a brief outline of current Army ammunition resupply doctrine, an overview of two of the most current Army models, and an examination of several past logistics theses done at the Naval Postgraduate School.

Chapter III presents the Logistics Model which is the primary focus of this thesis. It provides an overview of

model functions and explains the major model assumptions. Topics discussed include: the battle scenario, requests for resupply, the distribution of resupply assets, and the redistribution of on-hand assets.

Chapter IV discusses lessons learned in the construction of the model and outlines future areas of consideration for model enhancement.

Appendix A provides an expanded explanation of the material mentioned in Chapter III. This section, however, presents the material in a form suitable for the reader who is familiar with both combat modelling techniques and SIMSCRIPT II.5.

Appendix B is a detailed explanation of the model code itself. It provides the reader with definitions of all entities, sets, attributes, and variables used in the model, along with a listing and line by line explanation of the computer code written.

II. STATE OF THE ART

A. INTRODUCTION

This chapter presents an overview of the background research that influenced the formation of the ammunition resupply model. This information is provided both as a ready reference for the works investigated and as a means of explaining the underlying rationale of many of the concepts later adopted for use in the model. This chapter includes: a brief explanation of the current and future Army doctrine regarding ammunition resupply; a discussion of two of the Army's most up to date ammunition resupply models, HELAPS II, and ARM; and an in depth look at past logistics thesis efforts pertinent to this model.

B. AMMUNITION SUPPORT IN TODAY'S ARMY

Without sufficient ammunition resources a combat unit's effectiveness is degraded and the tactical alternatives available to its leaders severely restricted. For this reason, a procedure, which first seeks an internal solution through the redistribution of assets, and then an external solution through resupply, is standard regardless of unit

type. For example, at platoon or company level the redistribution of on-hand ammunition assets is standard operating procedure at the conclusion of any move, regardless if the move is offensive or defensive.

When ammunition becomes a dwindling resource on a weapon the platoon leader is notified, regardless if the weapon is an M-16 or the main gun of an M1 tank. The platoon leader then takes appropriate action either by redistributing his own platoon resources or requesting resupply from the company commander, or both. In a similar fashion, platoon shortages are examined at the company level and if necessary a resupply request is submitted to battalion. At battalion, a support platoon receives requests for resupply from the company then issues ammunition from the battalion's assets. This support platoon in turn replenishes its stocks from either the brigade ammunition transfer point (ATP) or the division ammunition supply point (ASP). The ATP and ASP in turn are supported by a corps storage area (CSA). A graphic representation of a divisional ammunition support structure is given in Figure 1.

The logic that dictates the resupply activity which should take place and where it should take place is the

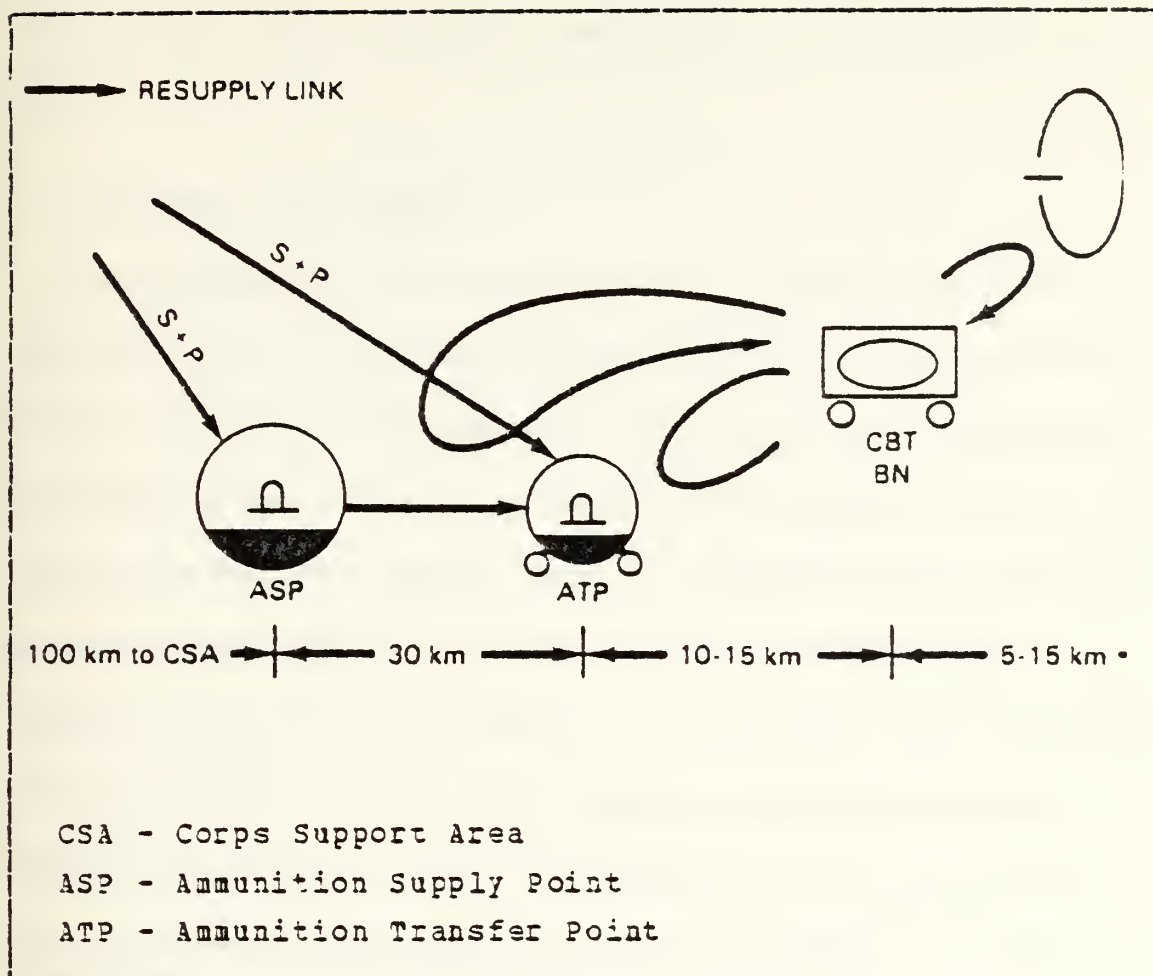


Figure 1: Ammunition Support Structure

objective of this model. Determination of the correct course of logistic action at each level of control from the individual weapon up is a function of how much is known about the situation at any time. This knowledge is admittedly imperfect due to time delays, human error, and the stress of battle. To capture the essence of this imperfect information flow, a central concept, Level of Need

(LON), was developed to mark thresholds for different activities. This concept is explained in detail in Chapter III.

C. EXISTING ARMY MODELS

The purpose of this section is to present the reader an overview of the two most recently developed Army ammunition resupply models. The two models are the Human Engineering Laboratory Ammunition Point Simulation (HELAPS II) and the Ammunition Resupply Model (ARM). Although other ammunition resupply models exist, the two mentioned above depict resupply at a level and degree of resolution that directly relates to this thesis. The purpose, characteristics, assumptions, input, output, strengths, and weaknesses of each model are discussed to provide an overview of the ammunition resupply modelling capabilities that are available today. The information presented in this chapter has been extracted from the referenced literature regarding the respective model.

1. HELAPS_II

The initial model for discussion is the HELAPS II model. This model was developed by Armament Systems Inc., Anaheim, Calif., under contract to the US Army Missile and

Munitions Center and School, Directorate of Combat Developments, Redstone Arsenal, Alabama. HELAPS II is a stochastic, event sequenced simulation that is run on a CDC 6000 series machine with GASP IV simulation language. The model simulates the internal operation of an ammunition resupply activity (CSA/ASP/ATP) to include the transportation system connecting the resupply point with its source of supply and its customers. The model constrains the flow of munitions through realistic delays representing the limited capability of material handling equipment (MHE), resupply vehicles (RSV's) and personnel operating in an environment susceptible to delays due to distance, time, weather conditions, integrated battle, and limited resources. [Ref. 1]

a. Purpose

The HELAPS model is meant to be used as an analysis tool for evaluating ammunition resupply activity dynamics, operational concepts development, TOE design/validation and mission area analysis. HELAPS II does this by simulating the movement of resupply vehicles (RSV's) from the customer or source element through the inprocessing, loading/off-loading operations, outprocessing

activities, and then back to their home elements. These activities all occur based on a workload generated in a realistic combat scenario.

b. Characteristics

The model uses a dynamic, computer, force-on-force wargame (ie. JIFFY) to determine the interval between resupply convoys by generating expenditures. These resupply convoys of up to 15 vehicles are constrained by availability of vehicles, distance traveled, enemy attack and environmental conditions. When the convoy arrives at the resupply point the RSV's are subject to delays caused by queues, processing, MHE/personnel availability, stock shortages, enemy attack and environmental conditions. Reliability, availability and maintainability (RAM) of equipment, along with internal resupply activity policies are also considered where applicable. Once all elements in a convoy are out-processed the convoy returns to its point of departure.

Throughout the running of HELAPS II information is collected concerning personnel activities and equipment performance. This information is analyzed to determine the following:

- (1) MHE/personnel utilization and performance.

- (2) Inprocessing/outprocessing delays.
- (3) Capacities for receipt and issue of munitions.
- (4) Optimal stockage objectives.
- (5) Effects of enemy actions on performance to include:
 - (a) Physical layout.
 - (b) Security requirements.
- (6) Distribution of resupply turn-around times.
- (7) Optimal mix of MHE and RSV's.
- (8) Impact of RAM on mission performance.
- (9) Effectiveness of operating procedures.
- (10) Adequacy of current or proposed TOE's.

c. Assumptions

All models make assumptions that play an important role in the results generated and analysis performed. The assumptions made in the HELAPS II model are:

- (1) Resupply requirements for individual weapons are consolidated at the firing unit's battalion.
- (2) All resupply activities take place on a 24 hour basis.
- (3) Inclement weather, night operations, and enemy suppression degrade performance of the resupply activity.

d. Input

Input data needed to run this simulation consists of numerous user entries. Some examples of this input data are:

- (1) Distances between firing units and resupply activities.
- (2) Distribution of ammunition consumption by firing unit for the simulation period.
- (3) Number of personnel available by duty position at the resupply activity.
- (4) Amount of stock by type available at the resupply activity.
- (5) Assignment of MHE to specific tasks at each resupply activity.
- (6) Distances between inter-activity elements.
- (7) A scenario of enemy activity.
- (8) All start, stop, and pause times.
- (9) Operating procedures for each resupply activity.
- (10) Environmental conditions by activity.
- (11) Host nation role if applicable.
- (12) Stockage levels for each ammunition type.

e. Output

This input data will generate output data of the following nature:

- (1) Amount of ammunition received and issued at a resupply point by type.
- (2) Start and stop stockage levels by ammunition type.
- (3) Discrete and average turn-around times by RSV/convoy.
- (4) Discrete and average processing times both in and out of the resupply point by RSV/convoy.
- (5) Discrete and average nonavailability times by equipment type.

f. Strengths

The major strengths of HELAPS II are its ability to accomplish the following:

- (1) Provide a tool to analyze the internal operations of a munitions resupply activity at any level.
- (2) Provide inferences as to the total issue capability of a designated TOE.
- (3) Collect data on the number of RSV's and convoys required to support a unit in a given scenario.
- (4) Provide refined estimates (ie. mean and statistical distribution) of the time required for an RSV or convoy to resupply a supported unit.

- (5) Identify the major choke points of a resupply activity and simulate the processing of each vehicle through these choke points in sequence. These choke points and the sequence in which they occur are as follows:
- (a) RSV/convoy leaves resupply activity.
 - (b) RSV/convoy is processed by Ammunition Supply Officer.
 - (c) RSV/convoy arrives at resupply activity and is in-processed.
 - (d) The RSV/convoy moves to its respective loading point.
 - (e) Loading/off-loading takes place.
 - (f) RSV/convoy leaves and proceeds to an assembly area.
 - (g) RSV/convoy out-processed through activity office.
 - (h) RSV/convoy returns to initiating activity.
- (6) Give a good evaluation of the effect of enemy activity on mission performance by considering the damage and destruction of support equipment.
- (7) Provide a good tool for Combat Service Support Mission Area Analysis and TOE development.

- (8) Provide another check on performance of the army's MHE design standards for handling munitions.
- (9) Look at scenarios under different environmental conditions.
- (10) Provide a tool for choosing the optimal location of resupply activities at all levels.
- (11) Identify security shortcomings for different scenarios.

g. Weaknesses

The major weaknesses of the model are:

- (1) Munition consumption and threat data bases must be developed manually from a force-on-force scenario.
- (2) The model requires extensive computer core memory allocation.
- (3) The model is expensive to run.

2. ARM

The second model to be analyzed is the Ammunition Resupply Model (ARM). ARM is an interactive, event oriented, time sequenced, computer model developed to simulate the various functions associated with ammunition resupply from the Corps Storage Area (CSA) down to the individual weapon. It was developed by the Combat

Operations Analysis Directorate (COA), CASSA, Fort Leavenworth, Kansas in March of 1980. ARM is written in FORTRAN IV and consists of approximately 3400 lines of code that require 144K octal main storage, including the data base. Less than 12 CPU seconds are used in processing the resupply actions that result from four hours of combat by a division size force. [Ref. 2]

a. Purpose

ARM's primary role is to assess the capability of a given TOE structure to respond to the logistical demands placed on it by various ammunition expenditures. For model purposes, these expenditures are created by a force-on-force model such as the JIFFY wargame. ARM was used in this manner to study the ammunition resupply capability of alternative division organizations of the Division 86 force structure. The model was developed in a general form thereby allowing it to be used with most combat models.

b. Characteristics

ARM controls the flow of ammunition through a network based on the capacity of a given number of RSV's and supply points. It also controls the flow through the supply

network based on MHE availability. The model actually forces the network to replace rounds at individual weapons at the unit level, and sends trucks back to designated resupply points to fill up and return. The functions of each truck are broken up into a series of discrete events portrayed as subroutines. The model takes each truck through a series of these subroutines (with operational availability and interdiction considered) in which actions are completed and times accumulated. Helicopter resupply, interactive command decisions, and tactical realism can be incorporated into the model.

c. Assumptions

A list of the key assumptions made in ARM are:

- (1) Only high volume/high demand ammunition is addressed.
- (2) Artillery units reload once each hour during a battle.
- (3) Aviation units reload upon the return of an aircraft.
- (4) Ammunition trucks are dedicated to a specific type of ammunition.
- (5) When a weapon system is lost, all ammunition is lost.
- (6) When a loaded truck is interdicted, the load is lost.
- (7) Helicopter emergency resupply will support only 155mm artillery batteries and the resupply originates at the ASP.

- (8) Trucks are sent for refill only when empty.
- (9) The division portion of the corps heavy lift helicopters will not exceed 10 CH-47's.
- (10) The division portion of corps transportation assets for ammunition resupply will be one medium truck company. This company will haul ammunition directly from the CSA to the ATP's and ASP's.

d. Input

The input parameters needed for the operation of the model are:

- (1) Number and allocation of ammunition dedicated RSV's/convoys in the transportation net.
- (2) Record of on-hand ammunition from the scenario wargame.
- (3) Number of weapons and basic load of each type system.
- (4) Ammunition hauling capacity of each RSV.
- (5) Stockage level and loading times at each ASP/ATP.
- (6) Key RSV characteristics (ie. speed, RAM).
- (7) Reload times of each weapon.

e. Output

Output from the model consists of an audit trail of all events accumulated in a series of reports generated

through a subroutine called REPORT. The following is a list of output reports available:

- (1) Status of each RSV to include location and load.
- (2) Current ammunition status of each unit.
- (3) Status of each ATP/ASP to include number of MHE available, RSV's in queue and the amount of each type ammunition on-hand versus initial stockage.
- (4) An interdiction report to include which RSV's were interdicted and the type and amount of ammunition lost.
- (5) Emergency resupply information.

f. Strengths

The major strengths of ARM are:

- (1) ARM influences the gamer's tactical decisions for ammunition logistics by adding scenario generated constraints.
- (2) The model can be used to determine what transportation assets of the ASP/ATP are required to support a given scenario.
- (3) ARM models individual RSV's.
- (4) The model gives an indication of the capability of a given ammunition resupply system for a given scenario.

g. Weaknesses

The principal weaknesses of ARM are:

- (1) ARM does not model the internal operations of the ASP/ATP in enough detail.
- (2) The model uses estimated straight line distances between firing unit and resupply points instead of actual road distances.
- (3) The model requires a manual data base development for each scenario.

D. LOGISTICS MODEL DEVELOPMENT AT NPS

The purpose of this section is to examine and discuss four previous NPS thesis efforts which modelled aspects of logistics. Three of these efforts are directly related to the STAR model.

Since the development of the STAR model at NPS, logistics planners have been given a unique opportunity to investigate the dynamic interaction between the combat and the logistics process at the most critical juncture, the battlefield. Since its initial coding in 1978, the model has been continually expanded to include an ever widening horizon of combat functions. Although several attempts have been made to integrate logisitics into the model, little

logistics is currently played. This section will review several previous logistics theses written at NPS. Some address STAR directly; others do not. In summary, these works form part of the evolutionary development of logistics modelling at NPS of which this model is a part. Discussion of each effort will include: a general description; an enumeration of assumptions; a discussion of the modelling techniques developed; a discussion of strengths and weaknesses of these techniques; and an outline of the model utility. Some of the concepts developed in these efforts are used in this ammunition resupply model; others will be valuable for future efforts.

1. "Simulation and Analysis of Transport in Support of a Combat Unit" by John R. Kelley (1978) [Ref. 3]

This thesis parametrically analyzes the mission of the support platoon of a U.S. Tank Battalion. The stated objectives of the thesis were: to develop an overall logistics model that could be integrated into a battalion level combat simulation; to measure the Support Platoon's ability to move supplies under varying conditions; and to evaluate the impact of simultaneous forces on combat support operations.

In pursuing these goals, a Monte Carlo simulation of the ammunition resupply process was developed. The use of a Monte Carlo simulation to capture the interaction of forces on the resupply process was a step away from traditional network/pipeline analysis normally developed for logistics studies. The technique involves the identification and isolation of significant variables within a process, assignment of probabilities of occurrence to each variable, and replication of the process described by these variables in order to achieve an expected value outcome. Using this technique, the author was able to focus attention on specific aspects of the resupply process, to deliberately vary values for the probability of their occurrence, and to statistically analyze results for significance.

a. Assumptions

The major assumptions made by Kelly in his thesis are:

- (1) The ability of a support platoon to provide ammunition support to the battalion is a direct function of:
 - (a) The maximum number of trucks available at any time
(Drivers assumed always available).
 - (b) The maintenance time required.
 - (c) The probability of ambush.

- (d) The probability of kill given an ambush.
 - (e) The loss of vehicles to ambush.
 - (f) The time required to replace lost vehicles.
- (2) Vehicles can make a maximum of three round trips a day to supply points.
 - (3) Maintenance readiness is evaluated at the end of each trip. Readiness is measured against a fixed operationally ready (OR) rate.
 - (4) Ammunition is obtained from a CSA located at the Division rear.
 - (5) All vehicles in a convoy are subject to enemy attack. Survival of each vehicle is measured against a fixed probability of kill. Partial damage and salvage are not played.
 - (6) Support is being provided to a "pure" tank battalion.
 - (7) The effects of each of the parameters measured are independent and additive.
 - (8) Measure of effectiveness selected: Truckloads moved during specified time periods. (3,5,15,30 days)

b. Method of Evaluation

Due to the software limitations on the analysis packages available when the thesis was written, the model

limited itself to a consideration of replacement time, probability of ambush, number of round trips delivered to supply points per day, and maximum number of vehicles available. Probability of kill given ambush, and probability of a vehicle becoming non-operationally ready were fixed for each simulation run. The impact of these two factors was combined into the mean of the design.

Having specified those aspects of the resupply process critical to the success of the Support Platoon mission, a range of probabilities for success for each of the critical parameters was fixed and a number of Monte Carlo simulations conducted for each variation. An analysis of the results was performed and an expected value for each set of selected probabilities was computed. The analysis conducted also measured the effects of each individual component, and its interaction with the other elements.

As a final demonstration of the methodology, a conflict set in a European scenario was developed and a simulation conducted. Values for each of the four critical parameters were varied in response to the scenario conditions, and in accordance with the judgement of the writer. A regression performed on the results fitted a

linear model to the data to check the linear fit of the postulated model. Parameters again were varied, results tallied, and tested for significance.

c. Strengths

The study achieved its objective of formulating and exercising a resupply simulation within the bounds of those parameters hypothesized as being critical. Accepting the premise that the factors modelled captured the critical essence of the resupply process, the technique used in the thesis could be modified and used in a generalized combat model to return a value of total tonnage of supplies delivered during a specified time period.

The primary conclusion of the analysis conducted, that the probability of enemy attack and the physical location of the resupply point were the driving forces behind the model, dictates that such parameters be of prime importance to any resupply model.

d. Weaknesses

The major weaknesses found in the model are:

- (1) Conceptual limitation to those parameters defined as critical. Use of parameters other than those specified and tested in the thesis would seriously weaken the

conclusions made in the thesis and could lead to incorrect results.

- (2) Current doctrine to include the creation and use of Ammunition Transfer Points (ATP's) was not considered.
- (3) Operationally Ready (OR) rates are normally determined on a daily basis rather than after each supply trip as proposed.

e. Utility

This model could best be utilized in the following manner:

- (1) The method developed and the results generated by the model would best be utilized as a generalized constraint to a high resolution combat model or, after some data analysis, as logistics coefficients in a Lanchester model.
- (2) The major finding of the model was that distance and probability of enemy activity were the driving parameters in the determination of overall mission accomplishment. As such, development of any resupply model should consider those parameters identified as critical in their structure.

2. "Simulation of Tactical Alternative Responses" by W. S. Wallace and E. G. Hagewood (1978) [Ref. 4]

This thesis and the follow-on work conducted by Wallace and Hagewood after their graduation from NPS formed the basis of what is now the STAR model. This original work was designed as a high resolution, event sequenced, stochastic model of ground combat. The language used was SIMSCRIPT II.5. Since much of what was developed is still an integral part of the present model, this document is still a vital reference tool. In building the simulation, logistics effects were modelled in three ways. These were:

- (1) The use of logistics as an engagement constraint by asking the question, "Do I have this ammunition on-hand to fire?"
- (2) The use of logistics as a time constraint by asking the question, "Given that I have this ammunition on board the tank, what storage compartment is it in, how long will it take me to access it and move it to the ready rack?"
- (3) The modelling of logisitical methods to create supply caches in order to resupply combat elements.

Of these three attempts to model the effect of logistics on the combat process, only the first, logisitics as an engagement constraint is currently used. The use of logistics as a firing constraint was abandoned after the XM1 stowed load study was completed. This constraint could easily be added to the existing STAR model but at the present time such a high degree of resolution is inappropriate. The modelling of resupply caches was abandonned because at that stage of the model's development, the contribution of the caches was of marginal value when weighted against the time required to prepare the data and the CPU time required to execute the logic. Tanks on the battlefield were killed, or the battle was ended before resupply was necessary and so the logic modelled became superfluous.

Each of the three attempts to incorporate logistics effects however, is worth analyzing in order to gain insight into the interplay of forces within STAR, and to gain modelling insight through access to existing working code.

a. Methodology

The general methodology of this model is as follows:

(1) Resupply As A Firing Constraint - in this logic, ammunition availability is played as a constraint to firing. The logic developed tracks on-hand ammunition by type for each tank. At the beginning of a firing sequence, a round is selected and its on-hand availability is screened on a go/no go basis. In implementation, ammunition is modelled at two critical junctures, when selecting the ammunition to fire, and upon round impact. The first juncture is controlled by routine PRIORITY.AND.ROUND.SELECT. This routine assesses the relative importance of the target currently selected, chooses a preferred round for use against the target, and checks to see if the stocks of the round are available. In performing this function, the routine accesses a matrix called a DANGER STATE array to determine the priority of the target, and to select the ammunition to be fired. Routine PRIORITY.AND,ROUND.SELECT then calls routine AMMO.CHECK which screens the ammunition attributes of the specific tank to determine if the ammunition is available. The second juncture occurs at the time the round is scheduled to impact. As part of the logic,

routine DECREMENT.AMMO is called to subtract one round of the type of ammunition fired from the tank which fired.

- (2) Resupply As A Limitation On Firing Time - the logic developed for this aspect of logistics was done in support of the XM1 tank stowed load study. It modelled the time required to physically move rounds inside an XM1 in an effort to add a degree of realism to the play by restricting the access to ammunition on the tank. This logic modelled the time required to move rounds from one of five storage compartments on the tank to the ready rack. This level of detail was later judged to be unnecessary in the normal use of the model.
- (3) Resupply - additional work after thesis completion attempted to extend ammunition concepts to include the movement of supplies from storage areas to resupply caches. The focus of this logic was a Supply Officer entity who controlled a number of caches in support of his unit. These caches are planned prior to program execution. In the execution, a routine PILE.SO.CREATE calls a number of subprograms which loads trucks at

the ATP (event LOAD.PLAN), moves loads to a pre-planned cache site (event MOVEOUT), and offloads the ammunition at the site (event OFFLOAD). Resupply of the combat vehicles (event UPLOAD) was accomplished when the unit withdrew to the location of the cache.

b. Strengths

The major strengths of the model are:

- (1) Logistics As An Engagement Constraint - the logic developed directly tracks the on-hand balance of 6 types of ammunition for each weapon system. This is a basic start point for any model that hopes to model logistics in STAR realistically.
- (2) Resupply As A Limitation To Firing - although this code proved to be of value in the XM1 stowed load study, the decision to turn off the code was more a function of lack of utility than absolute uselessness. The logic, in fact, provides an immediate tie-in for any future modelling of delivery of a resupply of rounds to the combat units.
- (3) Other logic developed depicts the loading and unloading of supply trucks and combat vehicles, creation of caches at pre-designated points, and a rudimentary stock control system.

c. Weaknesses

The weaknesses of this model were perceived in the following areas:

- (1) There was no overall logic developed to dynamically control and integrate logistics play within the model. Tanks will continue to fire until ammunition stocks are exhausted. Resupply caches are entirely pre-programmed and once execution begins, the dynamic flow of the battle will not change the creation of stocks.
- (2) The assets possessed by a unit will not influence the tactical decision process. Logistics influence is limited strictly to fire/no fire control.

d. Utility

The SIMSCRIPT coding developed is fully and immediately integratable into STAR and thus represents an excellent start point for any new logistics study. The basic STAR model interface points still exist. If future STAR logistics modellers understand this logic they will save themselves considerable time and effort.

The immediate extension of this logic includes recognition and use of the current ammunition status to

trigger resupply actions. Further extensions include the use of ammunition status to modify tactical courses of action or to trigger a request to move.

3. "A Dynamic Ammunition and Resupply Model in Support of the STAR Model" by Bruce G. Ripley (1979) [Ref. 5]

This thesis presents a conceptual framework for modeling logistics functions within a combat brigade. The objective of the thesis was to develop a logical structure for the modelling of ammunition and fuel resupply. It was hoped that this logical structure would lay a basis for the future development of detailed logistics logic to be entered into STAR.

The medium chosen to depict this framework was a network diagram. The primary product of the thesis was the development of this network and associated parameters essential to modelling resupply. Resolution of a fully developed model using this logic would depict individual trucks carrying rounds of ammunition from the brigade trains to the combat vehicles. Ammunition would be measured by the "box", a generic term used to represent all packaging configurations from the individual round to a pallet of

rounds. Petroleum would be measured in bulk terms only; packaged petroleum is not played.

a. Assumptions

The major assumptions made in this model are:

- (1) Battalion trains are located 5 Km and brigade trains are located 25 Km behind the FEBA.
- (2) Corps will transport ammunition from the Corps Storage Area (CSA) in the Corps Rear Area to the Ammunition Supply Points (ASP's) located in the Division Rear and the Ammunition Transfer Points (ATP's) located in each of the brigade areas.
- (3) Corps ammunition support to the division will be provided by two conventional ammunition companies with each company operating two ASP's.
- (4) Capabilities of ammunition points are as follows:
 - (a) ASP - Receipt and issue of 2000 short tons of ammunition daily.
 - (b) ATP - Receipt and issue of 500-600 short tons of ammunition daily.
- (5) Only 5 ton trucks are capable of making the round trip from the battalion trains to an ASP and back. AFARV's and GOER's are limited to trips to/from the ATP's at brigade.

(6) Vehicles will travel only in convoys.

(7) Operationally ready rates (OR) are computed once daily according to the following rates:

(a) Material Handling Equipment (MHE) - 75%

(b) Bn Ammo Vehicles - 85%

b. Characteristics

The resupply framework was developed around the traditional supply MOE of ascertaining the number of truckloads of ammunition delivered per unit time. In execution, the network traces the movement of trucks of varying dimensions through a network subject to uniformly distributed movement and loading times.

(1) Network Development - the key to the thesis work was the development of the network diagram itself. The network depicts the flow of supplies from division and brigade storage areas to the battalion trains. Central to this concept was the determination of the number and characteristics of carriers, arcs, and nodes which make up the system.

(a) Carrier: the term used to represent the actual supply trucks on the network. Each carrier type has specific weight and cube limitations. These

limitations are used to constrain a vehicle's capability to transport ammunition. The carrier is the critical focus of the system, since determination of the quantity delivered per unit time is based on the carrier's successful completion of trips to/from the supply points.

- (b) Arc: the term used to represent road segments on the ground. Arcs direct traffic flow and are used to determine straight line travel time between points. Load capacity of roads is depicted by limitations on vehicle speeds on individual arcs, for example, 10 mph over unimproved roads. Road congestion, although identified as potentially a major problem, was not explicitly modelled.
- (c) Nodes: these mark points of change within the network itself. Two types of nodes were specified, load state changes and travel state changes. Load state changes mark those locations where ammunition is loaded and unloaded. Activity at these points is modelled as time delays generated from specified distributions. Operationally, these delays would be modelled as a function of the

number of people at the point, the number of MHE pieces at the point, the capacity of the carrier, and the length of the queue at the resupply location. Travel state changes represent those points on the network at which direction of travel changes. These points represent travel through cities, past major intersections, and through points of congestion.

- (2) Movement And Load Times - a second key aspect of this framework is the concept of time use. All times within the model are assumed to be uniformly distributed about a calculated mean. Thus travel time along a stretch of highway is based on the length of the roadway and on the assumed speed of the vehicle. To this, an induced variability of plus or minus 20% was introduced to allow some further randomization of vehicle times. Load/unload times were based on an assumed loading time for the particular load on a specific cargo carrier. Thus, ammunition cargo varied by the "box" configuration to be handled, while fuel loading varied in accordance with the load capacity of the pump assumed to be at the location.

c. Strengths

The strong points of this model are:

- (1) The thesis successfully established a network diagram which adequately captures the various activities which make up the supply chain. The example illustrated in the thesis is in generalized enough terms to be easily adaptable to any particular setup.
- (2) Although designed for a FORTRAN simulation, the descriptors used to model the flow of the network can be adapted to any simulation language. These critical descriptors and the information they convey are as follows:
 - (a) Arcs: length of road segment; type of road (unimproved, etc.); average vehicle speed on the road; amount of congestion on the road.
 - (b) Nodes: type (road junction, supply point, town, etc); average delay time expected.
 - (c) Carriers:
 - Ammunition Carriers - type, max cargo weight, operationally ready status; location; amount of cargo on board; type of cargo on board; unit.

- Petroleum Carriers - type; fuel carried; amount carried; gallon capacity; pump rate.
- (d) "BOX" of Ammo: weight of package; volume of package; number of rounds/box.

d. Weaknesses

Some of the major weaknesses perceived by the authors in the model are:

- (1) In briefly outlining assumptions, locations of support units were discussed in terms of fixed distances rather than as envisioned in the more flexible design doctrine called for. In fact, present doctrine calls for the battalion trains to normally divide assets between two locations, the combat trains and the field trains. Combat trains are located 5-7 Km behind the FEBA while the field trains are located 10-15 Km behind the FEBA. The composition of either is flexible; however, the bulk of ammunition supplies is normally maintained at the field trains. Brigade trains are normally located 15-25 Km behind the FEBA and so could conceivably be co-located with the battalion trains. The central considerations which dictate the location of these facilities is the

mission of the unit and the range of enemy medium artillery.

- (2) The vehicle load times for various ammunition types were established for illustrative purposes. More realistic distributions must be developed for actual applications.
- (3) No provisions were made to model hostile action on the network.
- (4) No queue was explicitly established at any supply point.
- (5) The network was self-contained and designed to model only the movement of battalion trucks on the network.
- (6) Once a resupply vehicle arrived at the battalion trains, resupply was considered completed. No attempt was made to model the loading of ammunition on combat vehicles.
- (7) There was no decision logic developed to dynamically control the network or to react to a change command once the vehicles were set in motion.

e. Utility

The network designed is an excellent starting point for the development of resupply logic at the

battalion/brigade level. The detailed development of the essential elements of this network is a first cut at molding the disparate elements of resupply into a coherent process.

4. "A High Resolution Integrated Combat and Logistics Model" by D.G. Kirby and D.P. Schultz (1980) [Ref. 6]

This thesis was the first attempt to integrate the effect of logistics in the STAR model. The objective of the work was to design a broad flowchart of the programming logic required to model logistics and to investigate a means of modelling the command and control decision processes which overlay this process. The authors limited their discussion to the resupply of petroleum and ammunition. In depicting this development, the authors utilized the Software Decision and Documentation Language (SDDL) which outlines the logic of the program in the form of a detailed flowchart.

- a. Assumptions

The major assumptions made by Kirby and Schultz are:

- (1) The Battalion Support Platoon is solely responsible for battalion resupply. Ammunition is obtained from

either the ATP run by DISCOM, or from the ASP run by corps. Petroleum is obtained from DISCOM tankers spotted in the Brigade Supply Area.

- (2) Resupply can be accomplished either by moving supplies forward to the combat vehicles (unit resupply), or by pre-positioning ammunition at specified locations (Cache).
- (3) Supply vehicles carry homogeneous loads. No cross-leveling of cargo between supply trucks is permitted.
- (4) Ammunition will not be redistributed between elements of a unit.

b. Methodology

The logic designed can be divided into three categories:

- (1) Command logic within a battalion.
- (2) Unit resupply logic.
- (3) Supply point resupply logic.

The critical development by the authors was the design of the command decision logic; flow for the other two categories was relatively transparent. Command logic is used primarily to evaluate the current supply situation and to

determine a priority for resupply. The key to this logic lies in the development of the concept of Level of Need (LON) .

Level of Need (LON) is a categorical structure through which the resupply situation of a unit is expressed. It represents the ratio of supplies on-hand to the total capacity of the unit. The authors define four Levels of Need; full, want, approaching critical, and critical. LON is computed for each firing system with regard to its primary ammunition and its fuel status only. The lowest category computed for either determines the overall LON for the weapon system. At the platoon, this logic models the platoon leader examining the LON of each of his vehicles. The platoon is then assigned an LON based on the category it falls under. This platoon logic is duplicated at each company, battalion, and brigade in the resupply chain thereafter.

Decision logic for resupply uses this LON and combines it with a consideration of supplies available for issue and an evaluation of the suppression level at the unit being resupplied. A listing of all requests is then prioritized in accordance with the above criteria. Ties are

broken in favor of the unit with the greatest number of weapon systems alive, the thought being that maximization of available combat power is a commanders first concern.

c. Strengths

The thesis clearly outlines those essentials necessary for the modelling of resupply. Although no code was developed, the flow diagram developed illuminates the path to be taken. Decision logic is always a difficult area to model. Development by the authors of a conceptual basis for this process is invaluable. By specifying the urgency of need through the concept of LON the authors made the resupply decision logic workable. This procedure for prioritizing resupply efforts based on the factors enumerated outlines a clear and realistic model of the commander's thought process.

d. Weaknesses

Presentation of a combined LON depicting the fuel and ammunition situation is unrealistic. The need for ammunition and fuel must be assessed separately as each impacts on the tactical situation in a different manner. In fact, a further sub-division within these categories as to type of fuel and type of ammunition would present a clearer

and more realistic picture upon which to base tactical decisions. The method of combining LON at platoon and above based on a predominant LON and on a subjective assessment of the relative importance of tactical systems is also unrealistic. Again a sub-division of information into types of ammunition and types of fuel needed would add a clearer and more realistic dimension to the problem play. Failure of the authors to develop logic for the redistribution of on-hand assets is unrealistic. Addition of such logic would present a truer picture of the real process within units. Lastly, consolidation of ammunition on-hand at the platoon level and above must include a consolidated count of all assets on-hand, including reserve assets. Failure to do this would again cause decisions to be based on unrealistic data.

e. Utility

This thesis illuminates the path of development for future logistic modelling in STAR. The flowcharts presented are detailed and thorough. They totally explain the resupply network. This concept of modelling the decision framework overlaying the supply network, while needing significant revisions, steers future efforts in the right direction.

III. MODEL DESCRIPTION

A. INTRODUCTION

This chapter presents an overview of the ammunition resupply model developed for this thesis. It discusses each of its parts in general terms and lists its major assumptions. A detailed discussion of the model, to include an explanation of the SIMSCRIPT II.5 programming language and the model code developed is presented in Appendicies A and B.

The resupply model presented in this thesis is a stochastic, discrete-event simulation implemented in the SIMSCRIPT II.5 programming language depicting ammunition resupply procedures within a combat battalion. The basic structure of the ammunition support flow is taken from Army Field Manual 9-6, Ammunition Service in the Theatre of Operations [Ref. 7]. The model is designed to provide a flexible framework within which the user may specify the Tables of Organization and Equipment to be played and the critical resupply levels which will result in a resupply action. In its present form the model can play an unlimited

number of weapon systems and ammunition types, however, each individual system is limited to a maximum of 6 ammunition types.

Section B of this chapter discusses the battle used in generating the data for the model.

Section C explains how ammunition expenditures are tracked from the weapon to battalion and how such expenditures thereby trigger resupply action by the chain of command.

Section D discusses the resupply logic used at each level of command in evaluating the availability of on-hand ammunition and determining both the quantities released and priority of resupply.

Section E explains how the redistribution of on-hand ammunition assets is modelled and when it takes place.

B. THE BATTLE

The purpose of the battle in this model is to generate requirements that will force a response from the ammunition resupply logic developed. Initially, the authors intended to use the STAR model as a source for input data, since a record of ammunition expenditures is normally generated as part of its output. In the present configuration of STAR,

however, the battle terminates well before ammunition resupply ever becomes critical, so a direct tie-in to the model was deemed impractical. An alternate proposal for generating data from multiple STAR runs was also rejected as too cumbersome.

Instead, it was decided to generate the needed data through the use of Lanchester equations and Monte Carlo techniques. Admittedly, considerable realism and resolution are lost in doing this, but the simplification permits the generation of necessary data without the use of a prohibitive amount of computer time exercising the STAR model. It is important to keep in mind throughout the model that the battle generated is unrealistic and is used solely as an expedient to generate data for the logistics model.

Examples of the types of data generated by the battle for use in the resupply model are:

- Ammunition Expenditures Over A Given Time Period - this data is generated for each weapon system in the battle and each ammunition type it might possess.
- Damaged And Destroyed Vehicles - combat and resupply vehicles are periodically checked for battle damage by evaluating random number draws against a set of damage

probabilities assigned by the user to each type of blue vehicle in the model.

- Movement Of Units - within the present model movement of company units is accomplished on a random basis. The purpose of such moves is solely to execute the model's redistribution logic.

The major assumptions underpinning the battle's architecture are:

- Battles are fought for 6 hours a day. The start time of the battle is determined by a random draw.
- Each weapon type of a weapon system has a rate of fire assigned through user input. However, the same weapon on a different system can be assigned a different rate of fire if the user so desires.
- Ammunition expenditures are generated in the model by evaluating random number draws against weapon system probabilities of fire for each armament. Expenditures are then computed by multiplying the rate of fire for that armament times the elapsed battle time for that particular day.
- Combat vehicle damage is assigned on a random basis over four types of kills : firepower kills, mobility kills,

mobility/firepower kills, and catastrophic kills. For all types of damage except firepower kills, ammunition on board that vehicle is considered lost. Ammunition assets belonging to a vehicle that sustains a firepower kill are assumed undamaged, and are immediately redistributed to the other fighting vehicles within that fighting system's respective platoon.

- Movement is restricted to company units and can take place only after that day's battle.

C. REQUESTS FOR RESUPPLY

Expenditures of ammunition by individual weapons form the basis of resupply activity in the model. Key to this process is a concept called level of need (LON). A level of need is evaluated for each ammunition carried by a combat entity in the simulation. These individual vehicle LON's are aggregated to form LON's for each platoon and company in the model. The purpose of the LON is to provide a measure of the urgency of need a weapon or unit has for a particular ammunition type. An entity's LON is updated over uniformly distributed time intervals independently of other vehicles or units in the model. This technique was implemented in order to capture some sense of the imperfect information

that is inevitably generated and passed in any supply system. The purpose of this section is to explain this resupply request process and to discuss the effect of and reaction to the imperfect information by the chain of command.

1. Level of Need (LON)

Level of need is a concept that was originally developed by Kirby and Schultz in March of 1980. The basic idea they developed is adopted for use in this model with some major alterations. The concept of a level of need was adopted because it represents a single unifying idea that will allow the expansion of this model to all classes of supply.

Level of need describes the urgency of need a weapon has for a particular type of ammunition. This urgency is then sequentially passed to and evaluated at each level in the chain of command until a level is reached which can respond appropriately. A separate LON is computed for an ammunition type at the weapon, platoon, company, and so on with information from each lower level being fed into the computation of the LON of its immediate superior. In effect, this forces each level to respond to the battle flow.

The actual value for an LON is assigned based on the measured percentage of fill (amount on-hand / base load) of an ammunition type at a particular moment. The essential idea is to have threshold values for the percent fill of an ammunition type which will trigger a leader's actions. The benchmark for this computation is called a base load for that ammunition type. An LON continually charges as individual weapon entities, each with its own ammunition configuration, are damaged or destroyed.

At the weapon level, base load is equal to the initial stowed load for the particular ammunition in the weapon system. This load can be different for each weapon system within a platoon. An ammunition's base load, for a platoon, is equal to the sum of all stowed loads of alive systems in the platoon possessing that ammunition type. A company's base load is, in turn, determined by summing over the base loads within its platoons and so forth.

Level of need within the model is divided into 5 categories, the thresholds of which are controlled by user input. These 5 categories are defined as follows:

- a. Full ("5") - a weapon or unit has enough of its base load of an ammunition on-hand that no resupply is warranted for that ammunition type.

- b. Want ("4") - the weapon or unit's on-hand load is below full, but is not in a position to jeopardize the mission. Reaching a "WANT" LON initiates a resupply action at the lowest priority.
- c. Approaching Critical ("3") - this level implies that the weapon or unit's on-hand ammunition is at a level warranting a higher urgency of need for resupply and a greater priority for fill when ammunition becomes available than the "WANT" level.
- d. Critical ("2") - the weapon or unit's on-hand ammunition has reached a level that seriously endangers mission accomplishment to the point that survival of the weapon or unit may become a problem. Immediate action is essential.
- e. Empty ("1") - the weapon or unit has no on-hand balance for a particular ammunition type and is no longer able to perform its mission.

Weapon and unit LON thresholds for the above categories are left as user inputs and must be supplied by the user for each combination of ammunition type and level of command. A tank then has different threshold values for the several ammunition types it carries. This corresponds

to placing degrees of importance on types of ammunition. So, while a tank might be considered critical for APDS rounds when it reaches 30% of its stowed load, it might not become critical for 50 caliber ammunition until it reached 10% of its stowed load. At the platoon, the threshold values for these same ammunition types would be different from those of the individual fighting systems. This models a platoon leader's wider perspective on a battle situation. This situation is repeated at successively higher levels of command.

2. Requests for Resupply

The resupply process begins at the weapon system where ammunition status is periodically updated. The time between updates is determined based on draw from a user defined probability distribution. The platoon, for its part, periodically updates its own status by obtaining information from each of its assigned weapon types. The company, in turn, updates its status by obtaining information from each of its platoons. The information "passed" to each level is that obtained from each entity's most current update rather than from any source of "perfect" information. Requests for resupply below company level are

limited to those ammunition types for which a vehicle or platoon is empty. These requests alert the next higher level to update its ammunition situation and to take action appropriate to its level.

The formality of a resupply requisition is introduced at company level where resupply requests from the company to battalion are triggered every time a company's LON changes for an ammunition type. Quantities requested vary in accordance with the assets possessed by the requesting entity.

3. Imperfect Logistics Information

Information passed during a battle is approximate at best. The imperfect nature of this information is a result of many factors, including:

- a. Estimates of on-hand ammunition made at the weapon during combat - It is frequently impossible to stop and count ammunition assets during the heat of an engagement; educated guesses are often the rule rather than the exception when passing ammunition information.
- b. Time lapses between resupply requests and delivery of requested material - From the time the request is forwarded until the delivery of the ammunition,

additional resources will be expended and weapon systems lost.

c. Simple counting mistakes.

Within the model, imperfection of the logistics information is induced as follows:

- Weapon systems update their ammunition LONs periodically at random times within user established minimum and maximum times. Upon request from platoon, the weapon system provides its most current count; that is, it provides the information obtained from its own last ammunition update.
- Platoons and companies can obtain their information only from their immediate subordinates, again at random times within user specified intervals. This procedure duplicates the periodic requests for ammunition updates from platoons and companies to their subordinates during a battle. Again, the information reported by each subordinate level is that obtained from its own last update.
- At the company level, formal resupply requests are created by ammunition type as an ammunition's LON value changes. This corresponds to a company periodically

reviewing and updating its resupply requests because of ammunition expenditures or loss of ammunition due to vehicle damage or destruction. The quantity requested each time is the quantity which would be required to bring a unit back to full base load. At the supply unit, upon receipt of a new resupply request, this new requisition is filed, and any older requests for that ammunition and that company are destroyed.

It is important to note that in both a and b above the modelling techniques described give the user the flexibility to make the logistics information flow as accurate or inaccurate as desired by controlling the randomness of the LON updates.

4. Assumptions

The major assumptions made during the resupply request process are:

- a. Requests for resupply are an iterative process up the chain of command with each level receiving information only from its immediate subordinates.
- b. An LON of empty ("1") initiates an immediate request for action up the chain of command.

c. Formal resupply requests from the company are triggered by changes in an ammunition LON. The actual quantities of ammunition used to calculate the LON's are available at each level of command so that resupply can be affected at that level if appropriate.

D. RESUPPLY

The resupply process begins with the receipt of a resupply request by the Battalion S-4. This receipt initiates a sequence of events which ultimately results in rounds being placed on the weapon system itself. This section explains how resupply of requested ammunition is accomplished. The explanation includes modelling the Battalion S-4's decision logic; the resupply logic of the company after assets are received from battalion; and the distribution decision logic of the platoon leader after a resupply is received from the company. The explanations are general in nature. A detailed discussion of the logic is contained in Appendix A, Section D.

1. Battalion Distribution

A battalion's initial reserve of ammunition is determined by the number and type of resupply vehicles (RSVs) in the support platoon and the type and amount of

ammunition those vehicles are designated to carry. This information is determined by the user and entered as input. An RSV's hauling capability for an ammunition type is determined by its cube or weight limitations, whichever is reached first.

Within a battalion, the S-4 is responsible for controlling the distribution of battalion reserve ammunition assets to subordinate companies. The S-4's responsibilities include the following:

- a. Review and prioritization of requests filed by priority and time of request. The most critical LONs ("1") are filled first. If there is more than one, the requests are filled in the order they were received.
- b. Determination of the number of RSV's necessary and available for resupply missions.
- c. Determination of the proper mix of ammunition types to be delivered to a unit in the face of multiple requests and limited transportation.

2. Company Distribution

Upon the arrival of a resupply convoy, a Company Commander takes the following actions:

- a. Determines the type and quantity of the ammunition he has received.
- b. Determines the immediate needs of the platoons in the company.
- c. Distributes the ammunition received to each platoon in amounts dictated by their immediate urgency of need.
- d. Re-evaluates the company's levels of need.

3. Platoon Distribution

The Platoon Leader's responsibilities for the distribution of ammunition resupply to the weapons within the platoon are the same as those of the Company Commander.

4. Assumptions

The major assumptions made in developing logic to model a battalion's ammunition resupply distribution process are:

- a. When a resupply action is triggered at battalion, the S-4 responds only to those requests he has knowledge of and then only in the amounts listed on that request. No further update is permitted until a new request is received.
- b. RSVs will not be dispatched from the battalion trains with less than a half load of ammunition unless the load contains ammunition with an LON of "1".

- c. RSV's can carry loads of mixed ammunition but only to one company.
- d. RSV's resupplying the same unit will travel in convoy.
- e. If an RSV is damaged or destroyed all the ammunition it is carrying is assumed to be destroyed. The ammunition is not replaced during the simulation run.
- f. RSV's that complete a resupply mission arrive back at the battalion trains with the same load they delivered after an appropriate time delay. This simulates the RSV's round trip to an ASP/ATP and permits restocking of battalion ammunition assets.
- g. Ammunition stockage at the battalion trains is limited to the total weight and cube limitations of the battalion ammunition trucks earmarked to haul it. Individual RSV loads are not "fixed" but rather remain flexible, subject only to the stockage at the trains and the weight and cube restrictions of the vehicle itself.
- h. An ATP/ASP has unlimited supplies of all ammunition types. The only limiting factor on the amount of ammunition an RSV takes back to its battalion trains is the vehicle's own weight and cube limitation.

E. REDISTRIBUTION

In an actual combat situation redistribution of on-hand ammunition assets is performed as standard procedure in certain situations. This section describes the situations which warrant redistribution in this model. It explains when and where the redistributions occur and the major assumptions made in performing them.

1. Redistribution Due To Relocation

Redistribution takes place immediately upon completion of any unit move. This activity is accomplished within platoons only, with its objective being the even redistribution of all on-hand assets. In the model, the following actions take place when a move is completed:

- a. All weapons give their respective platoons an ammunition update.
- b. Each ammunition type is divided evenly among weapons using it with respect to the weapon's stowed load.

2. Redistribution Due To A Firepower Kill

Redistribution of on-hand assets is performed in the event of a vehicle sustaining an F-kill. In this case, the platoon redistributes the ammunition as if it has just received a resupply equal to the ammunition on the F-killed vehicle.

3. Assumptions

The major assumptions made during a redistribution are:

- a. Redistributions only take place at the platoon level.
- b. A vehicle receiving an F-kill becomes an RSV until all on-hand ammunition is distributed.

IV. CONCLUDING REMARKS AND FUTURE ENHANCEMENTS

A. GENERAL

This ammunition resupply model represents the first step toward the eventual development of a low level, high resolution logistics model designed to interface directly with a comparable combat model. Program logic thus far developed explicitly depicts current U.S. Army supply doctrine at the battalion level. Beginning with the individual firer, the model simulates the ammunition resupply network responding to identified needs and ultimately providing the appropriate ammunition to weapon systems.

In executing this process, the model performs the following functions: recognition of shortages at all levels; initiation of requests for resupply appropriate to the level of command; determination of quantities to be released to fill requests; and delivery of supplies down to the weapon systems. The authors have tried to keep the model as flexible as possible by designing it in a manner that lets the user assign values at input for the critical variables

of the resupply process. The remainder of this chapter is used to lay the foundations for continued work based on the ideas developed in this thesis. The approach taken in outlining the direction of future efforts is threefold: to explicitly highlight some of the model's major deficiencies; to discuss several possible development paths which might be taken in expanding the model; and to discuss adjustments necessary to integrate this model with a comparable low level, high resolution battalion combat model.

B. MODEL DEFICIENCIES

Fundamental to understanding what a model can do is the equally important issue of knowing what a model cannot do. This model is deficient in the following areas:

1. Battlefield Realism - Due to the simplified nature of the battle, combat processes are not well played. Basic forms of maneuver, elementary command decisions, and individual combat action are not modelled beyond the simplest levels. These activities have a significant impact on the supply system and represent a critical deficiency in this model.
2. Damage Assessment - The simplified damage assessment routine, limited to combat weapon systems only, was

developed solely to drive the resupply logic.

Extension of this damage logic to resupply vehicles, development of a maintenance recovery and repair capability, and an ability within the supply system to respond to item losses would be a significant gain.

3. Movement - The model does not depict movement beyond the imposition of a simple time delay for travel from one section of the battlefield to another. These delays are based on doctrinal distances and do not consider terrain, weather, or suppression. The addition of terrain and movement modules would add a significant dimension to the model and permit the extension of logic into related resupply issues such as route selection, traffic control, and traffic congestion.

4. Resupply Logic

- a. The battalion played in the model is always resupplied, after a time delay, with the exact type and quantity of ammunition it has released to its companies. The source of this resupply is an ATP/ASP that contains unlimited ammunition assets. These assumptions significantly reduce the realism

of the model and should be amended to include, at a minimum, a limit on the resupply available to a battalion dictated by the prevailing Command Supply Rate (CSR) and Required Supply Rate (RSR).

- b. Convoys in the model are limited to point to point delivery of ammunition. Multiple deliveries by one convoy to several companies is not allowed. This deficiency decreases the model's realism by limiting a battalion's ability to efficiently use its transportation assets, and a company's ability to control these assets when a convoy arrives.

C. PATH OF FUTURE DEVELOPMENT

Having initiated a basic structure for the resupply model, expansion paths for resupply logic, both within the confines of the battalion model itself, and beyond the battalion supply point to brigade have become apparent. The expansion ideas mentioned in this section will be limited to those areas of improvement within the supply system itself, purposely disregarding issues directed at the model's interface with a high resolution combat model. Some of the points mentioned in this section have been identified as model deficiencies but are re-mentioned here for emphasis.

Expansion of this model, with respect to resupply logic, is needed in the following areas:

1. Development of logic to more realistically depict the transport of ammunition from the ATP/ASP to the battalion trains. Such logic should include the explicit modelling of supply routes and the traffic control points overseeing these routes.
2. The effect of enemy interdiction efforts on the rear area supply points.
3. Development of a routine to extend the possibility of damage to resupply vehicles and convoys.
4. Development of routines to model maintenance, recovery, and repair as well as replacement of damaged systems.
5. Extension of delivery logic to allow resupply vehicles and convoys the option of delivering to more than one company.
6. Addition of movement and terrain logic.
7. Improvement of the redistribution logic to include provisions for an emergency resupply of ammunition if a situation warrants it.
8. Explicit representation of activities at the ATP/ASP to include queue and service times for trucks, and

ATP/ASP interface procedures. As mentioned in Chapter 2, much of this logic has already been modelled in HELAPS II .

9. The imposition of a Command Supply Rate (CSR) and Required Supply Rate (RSR) as a driver for the entire resupply process.

D. INTEGRATION INTO A COMBAT MODEL

Integration of this model into a low level, high resolution combat model presents the following problems: the problem of interfacing independently developed program logic; the necessity of developing additional command and control logic to blend the two models together; and the need to adjust the tactical decision making process to include the effects of logistics. Each of these issues is discussed separately below.

The problems of merging an independently developed resupply program into a fully developed combat model were recognized and carefully considered in the development of this ammunition resupply model. To overcome these problems, the model was developed as a self-contained module. The foreseeable interface problems with a combat model are thus limited to insuring that: the combat model captures

ammunition expenditures and passes them to the resupply logic; the resupply model has movement logic that interfaces with the movement logic of the combat model; and the supply model's resupply routines interface with the weapon systems of the combat model when delivering ammunition. Since the model developed in this thesis was specifically designed to interface with the STAR model, these changes would be limited to: the addition of several attributes to the chain of command structure; the addition of several attributes to the weapon systems; and the integration of convoy movement with the STAR movement logic.

Perhaps the greatest change caused by the addition of a resupply module would be its effect on the command and control logic of the combat model. A combat model's decision process could be expanded to include at least the two most basic methods of resupply for a battalion, unit and cache. Consideration of these two methods would necessitate development of logic which could dynamically answer the following questions:

1. Who has priority of resupply?
2. What ammunition is to be released subject to what command restrictions?

3. How will resupply be accomplished? Will supply personnel be present with any type of equipment?
4. Where will the resupply be accomplished? At the current location? At a subsequent position? At a rendezvous point?

Lastly, combat decisions are inevitably influenced by the availability or nonavailability of resupply. Inclusion of resupply logic into a combat model would force an active consideration of resupply issues in making the following tactical decisions:

- a. Adjusting rates of fire.
- b. Changing a weapon's primary ammunition of engagement.
- c. Decreasing or increasing a weapon's range of detection and engagement.
- d. Movement to alternate positions or the withdrawal of a unit.

APPENDIX A

DETAILED METHODOLOGY OF THE MODEL

A. INTRODUCTION

This appendix gives a detailed description of the ammunition resupply model in a format suitable for use by programmers and analysts. The discussion in this appendix approaches the model from a broad perspective in order to show the effect of the techniques used to model the "real world" resupply process. Prior to discussing the methodology itself, a brief description of the SIMSCRIPT II.5 programming language used in the model is provided for readers unfamiliar with the language. A detailed explanation of the actual code developed is provided in Appendix B.

B. USE OF SIMSCRIPT II.5 IN THE MODEL

The SIMSCRIPT II.5 programming language is designed to model discrete-event simulations. It is a user friendly language with a structure very similar to everyday speech. This feature enables a reader to quickly grasp and follow the flow of any program. Beyond the narrative clarity,

SIMSCRIPT provides an organizational structure which extends a programmers conceptual horizon beyond the normal bounds imposed by the use of variables and arrays. Central to this structure are the key ideas of entities, attributes, and sets.

Entities are program elements whose characteristics are being modelled in the simulation. In the ammunition resupply model for example, tanks, platoon leaders, company commanders and supply officers are classes of entities in the system. Attributes are descriptors which depict the entity's characteristics. In the model, every platoon leader entity carries attributes which define his unique company commander. Thus, although all entities in the same class have the same attribute names, they can be distinguished from each other by the values of their attributes. Attributes may have real, integer, or alphanumeric values.

A set is a collection of entities possessing some common property. The ammunition resupply model uses sets to track the type and amount of ammunition on-hand in each platoon. This is done by creating an entity for each type of ammunition with attributes which record the quantity

required and actually on-hand. Each entity is then filed in a platoon's ammunition set and its attributes thereafter track only that platoon's ammunition status.

An event in SIMSCRIPT is an occurrence which takes place at a specific simulated time. Events can change the values of entity attributes, remove or add entities to sets, create or destroy entities, or schedule other events to take place at later times. In the model, a weapon which expends all of its ammunition keys an event which notifies the platoon leader and starts decision logic for resupply. Events take place instantaneously and do not consume simulated time.

The use of SIMSCRIPT II.5 greatly simplifies the tracking of ammunition expenditures throughout the chain of command. The actual set, entity, and attribute structure used in this model is explained in detail in Appendix B.

C. GENERAL MODEL METHODOLOGY

The ammunition resupply model developed for this thesis is a stochastic discrete-event simulation designed to portray ammunition resupply procedures within a U.S. combat battalion. The model is a stand-alone, closed loop process which simulates the following activities within a combat battalion: periodic updating of individual weapon and unit

ammunition status; recognition of the need for and submission of requests for resupply; and receipt and issue of supplies from battalion reserve stocks. The overall process modelled is depicted in Figure 2 below.

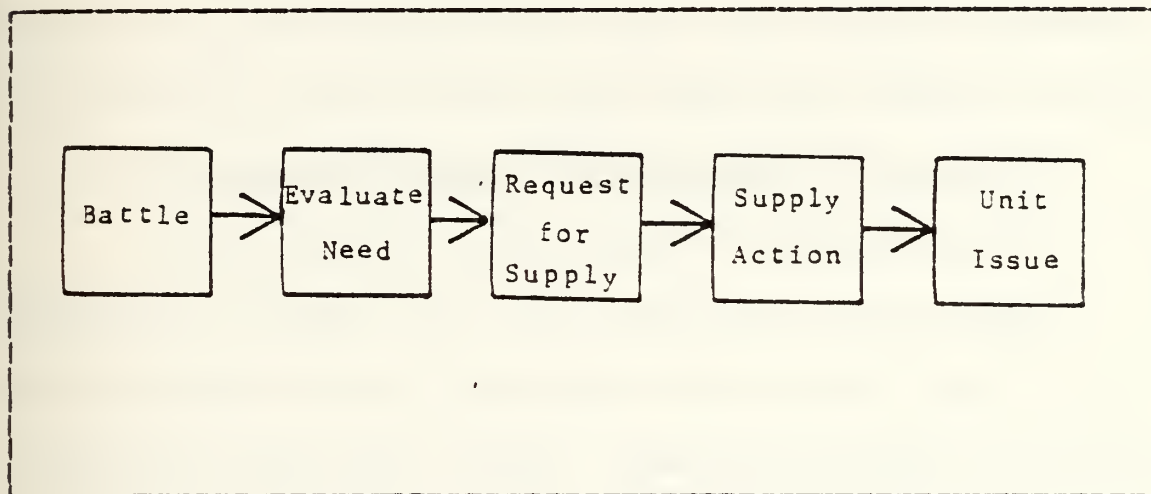


Figure 2: Resupply Process

The fundamental process depicted in the figure is duplicated at all levels of command (weapon, platoon, company, and battalion), differing only in the response options available at each level.

D. INPUT REQUIREMENTS AND THE INITIALIZATION OF DATA ARRAYS

Input to the model is used to accomplish the following: the creation of entities played in the simulation; the establishment of chain of command relationships between

entities; the scheduling of initial ammunition update times; and the establishment of parameter values for supply action and response(LON Thresholds). This initialization process is controlled by routine BLU.CREATE, which creates the major entities played and calls, in turn, routine BASIC.LOAD to establish initial ammunition loads, and routine PARAMETERS to initialize the critical data arrays and global variables.

1. Generating Resupply Requirements - The Battle

Generation of requirements to exercise the ammunition resupply model was initially to have been accomplished through interaction with the low level, high resolution STAR combat model. However, attempts to utilize the STAR battle in its current configuration led to difficulties with the pace of battle problem previously discussed in Chapter 1 and the objective was abandoned, at least for this thesis effort. In lieu of this, a simplified battle was developed strictly to generate requirements for the model. It must be emphasized that significant conclusions cannot be drawn from the battle summaries produced by this model. The sole function of the battle designed is to generate requirements in order to exercise the model's resupply logic. The requirements generated for

the model can be grouped into three broad categories: ammunition expenditures, damage and destruction of combat vehicles, and unit movement. The following paragraphs explain how the information generated is used in the model.

a. Ammunition Expenditures

Ammunition expenditures take place only during scheduled battle periods. These periods are randomly scheduled for six hours a day by the event BAT.L.TIME. Assessment of the quantities of ammunition fired by each weapon is determined in routine BATTLE which is called by each weapon system when it updates its ammunition status. In execution, routine BATTLE performs the following functions:

- (1) Checks if a battle is in progress - This is simply a check if the simulation time, TIME.V, is greater than the battle starting time, B.START, and less than the battle's end time, B.END. If it is outside this limit, there is no active on-going battle and the routine returns without action.
- (2) Determines if a weapon fires - A check is made on each of six possible weapons carried by a weapon system to see if they have fired. This is accomplished by comparing successive random number draws against a

probability of firing for each ammunition owned by the weapon system. If the random number drawn is less than the assigned probability of fire, the ammunition being tested has fired, and expenditures are computed. If not, no expenditures are computed and the logic transfers to the next ammunition carried on the weapon system.

- (3) Expend Ammunition - Each of the six ammunition types carried by a weapon system is assigned a unique rate of fire. This rate of fire is stored in an array, ROF, which is input in routine PARAMETERS. If the determination is made that an ammunition has been fired, the quantity expended is computed through the use of an exponential function. Lambda for the function is set equal to the ammunition rate of fire and the exponent is completed by multiplying this lambda times the elapsed battle time. This technique inevitably causes a greater expenditure of rounds as the battle time increases, however, its overall effect on the running of the model is negligible.

b. Vehicle/Weapon Damage and Destruction

Logic depicting the damage and destruction of vehicles and weapons was added in order to exercise the model's capability to adjust for ammunition assets lost throughout the battle. Losses and damage are generated only for combat weapon systems. Supply systems are not subject to combat loss or damage.

The modelling of vehicle/weapon damage and destruction is done in routine BATTLE with assessment of loss or damage limited to the 6 hour battle period. There are four types of damage played, M-kill, F-kill, M/F-kill, and K-kill. Probabilities for each type of damage are weapon system unique and obtained from an array, POD(Probability of Damage), input in routine PARAMETERS. Assessment of damage is accomplished by drawing a separate random number against each probable type of damage. If the number drawn is less than the probability for the type of damage being reviewed, the weapon sustains the damage. This technique leaves open the possibility that a weapon may sustain multiple types of damage, however, this side-effect is of little importance to the model's execution.

c. Unit Moves

The possibility of a company relocating during the simulation was incorporated in order to exercise redistribution logic contained in the supply model. The assumption underlying the need for the inclusion of this logic is that tactical units will seek to redistribute ammunition either before or after displacement in order to achieve an ammunition balance among its weapon systems. The execution of a unit move within this model in no way affects the conduct of the remainder of the battle. Unit moves are scheduled randomly in event BAT.L.TIME at the start of each 6 hour battle. Redistribution of company assets is accomplished upon completion of a move within each of the company's platoons. Cross-leveling of ammunition between platoons is not modelled. In execution, each company draws a uniform (0,1) random number, and, if that number is less than a user designated probability of move, the company will move at the end of the battle.

2. Determining the Level of Need (LON)

The determination of Level of Need is performed independently and at random intervals for every weapon system, platoon, and company played in the simulation. The

purpose of an LON is to provide an indication of the urgency of need an element has for each of the ammunition types it possesses. A distinct LON value is computed at each level of command. This models the increasingly wider perspective of the battle taken by commanders further up the chain of command. The net effect of this technique is to limit the importance of any one ammunition type as it is factored against other ammunitions played. The following subsections fully discuss the details of the process just described.

a. Imperfect Information

As discussed in Section C, Chapter 3, information in a logistics network is approximate at best. In the model, this imperfection is achieved by randomizing the times at which ammunition updates occur at each level and by limiting the knowledge passed from one level to another to that obtained in the most recent update. The following example illustrates the effect of this technique. A tank updates its ammunition status 10 minutes into the battle and finds 40 HE rounds on-board. At 30 minutes into the battle the tank has 30 rounds remaining. At this point, the platoon leader conducts an update and is informed that the tank has 40 rounds on-board. The platoon leader then

bases his decisions on this imperfection information. Similar update processes are performed at each level of command with the information provided forming the basis of LON computations at that level. The degree of imperfection in the information passed depends on the length of time between updates. These time intervals are entered by the user as input in routine PARAMETERS.

b. Initial Assets and Capacities

The initial ammunition assets of each weapon and resupply vehicle are input by the user in routine BLU.CREATE as the temporary attributes OH1 through OH6. The number of ammunition types that can be played in the model is unlimited; however, the quantity of each ammunition type on board a weapon system is limited to a maximum stowed load figure. Maximum values are set for each ammunition type and carried as the attributes, SLOAD1 through SLOAD6, on every weapon system. The stowed load configuration on a weapon system represents the users assessment both of what should be and what can be carried on a weapon system. This configuration is unique to each combat weapon system entity.

The platoon and company equivalent of a stowed load for an ammunition type is called the base load for an

ammunition type. Base loads for an ammunition type are computed by summing over the stowed loads carried on all undamaged elements within a unit.

c. Weapon LON's

Weapon systems form the basis for all LON calculations in the model since it is at the weapon level that ammunition is expended. Levels of need for a weapon system are computed in routine W.AMMO for each of the six possible ammunition types a weapon system might possess. Routine W.AMMO calls routine BATTLE to generate ammunition expenditures, and based on these expenditures, W.AMMO updates a systems knowledge of its current ammunition status. W.AMMO is called from several events for different purposes.

Event UP.W.AMMO calls routine W.AMMO randomly throughout the simulation in order to model a weapon system's crew periodically checking its on-hand resources. UP.W.AMMO is scheduled individually for each weapon system played based on successive draws from a uniform distribution. The delimiting times for the distribution, WMIN and WMAX, are input by the user in routine PARAMETERS.

The event is repeatedly re-scheduled throughout the simulation unless the weapon updating sustains some battle damage.

Events CO.RESUPPLY.ARR, REDISTRIBUTE, and FIREKILL call routine W.AMMO for all undamaged weapon systems within a unit in order to obtain an immediate update of the current situation prior to executing their respective program segments. These calls simulate a weapon system's crew checking its on-hand resources prior to any resupply action.

The actual calculation of an LON for an ammunition type is accomplished by taking the on-hand ammunition of an undamaged weapon and dividing it by the authorized stowed load of that ammunition for that weapon. The resulting percentage is compared to the weapon system threshold values stored in the array WPNLON which is initially input in routine PARAMETERS. The threshold values contained in the array mark the lower boundaries of LON categories. Figure 3 is an example of an LON calculation for APDS ammunition on board a tank.

System type - Tank

Wpn type - M1 $\%$ = on-hand / stowed load

Ammo type - APDS = 20 / 40

On-hand - 20 rounds = .5

Stowed Load - 40 rounds

WLON_THRESHOLDS (TANK,APDS)

"5" \geq .85

"4" \geq .60

"3" \geq .40

"2" \geq .15

"1" \geq 0.0

Therefore since .60 \geq .50 \geq .40

WPLON = "3"

Figure 3: Weapon LON Example

d. Platoon LON

Platoon LON information is updated in the routine P.CLASS.V. The process performed in this routine is essentially a summation of the information carried on-board the undamaged weapons in the platoon. This process updates

both the platoon's current on-hand information, and the base load information for that ammo type. As each ammunition is updated, a platoon percent fill is calculated for each ammunition type by dividing the current on-hand quantity of an ammunition by its current base load in that platoon. The percent fill calculated is compared to the platoon LON threshold values for that ammunition type. These critical values are stored in the array PLTLON which is input in the routine PARAMETERS. The threshold values contained in the array mark the lower boundaries of the lon categories. P.CLASS.V is called by several events for different purposes.

Event UP.PLT.AMMO calls routine P.CLASS.V randomly throughout the simulation in order to model a platoon leader periodically checking the platoon's on-hand resources. UP.PLT.AMMO is scheduled individually for each weapon system played based on successive draws from a uniform distribution. The delimiting times for the distribution, PMIN and PMAX, are input by the user in routine PARAMETERS. The event is repeatedly re-scheduled throughout the simulation unless all weapons in the platoon sustain damage and can no longer use ammunition.

Events CO.RESUPPLY.ARR, REDISTRIBUTE, and FIREKILL call routine P.CLASS.V for all platoons within a unit in order to obtain an immediate update of the current situation prior to executing their respective program segments. These calls simulate a platoon leader checking the unit's resources prior to any resupply action.

Figure 4 is an example of how a platoon LON is calculated. It is important to note in the example that the stowed load and on-hand ammunition of the 3rd tank is not considered because the tank is an M-kill. Also, the stowed load of tank 2 is disregarded because the weapon can no longer fire since it has been F-killed. The on-hand ammunition of tank 2 is not dropped from the computation however, due to the fact that the tank is still mobile and can be used as a resupply vehicle to deliver ammunition to other systems in the platoon.

e. Company LON

Company LON values are computed and assigned in routine COM.AMMO. In evaluation of this LON, the first step performed is to sum over all assigned platoons in order to update the company's on-hand and base load information.

	<u>SYSTEM</u>	<u>WPN</u>	<u>AMMO</u>	<u>STOWED LOAD</u>	<u>ON-HAND</u>	<u>STATUS</u>
(1)	Tank	M1	AP	40 rounds	20 rounds	Alive
(2)	Tank	M1	AP	40 rounds	15 rounds	F-kill
(3)	Tank	M1	AP	40 rounds	30 rounds	M-kill
(4)	Tank	M1	AP	40 rounds	25 rounds	Alive

PLT LON THRESHOLDS (APDS)

"5" ≥ .90

"4" ≥ .65

"3" ≥ .45

"2" ≥ .20

"1" ≥ 0.0

% = Tot Plt on-hand ammo(AP) / Sum Wpn stowed loads (AP)
 = 60 / 80 = .75

Therefore since .90 ≥ .75 ≥ .65

PLTLON = "4"

Figure 4: Platoon LON Example

A percent fill value is then computed by dividing the on-hand quantity for each ammunition by the required base load for that ammunition. This percent fill is then compared to company LON threshold values stored in the WPNLON array

which is input initially in the routine PARAMETERS. The threshold values contained in this array mark the lower boundaries of the LON values. COM.AMMO is called by several events for different purposes.

Event UP.COM.AMMO calls routine COM.AMMO randomly throughout the simulation in order to model a company commander periodically checking the platoon's on-hand resources. UP.COM.AMMO is scheduled individually for each weapon system played based on successive draws from a uniform distribution. The delimiting times for the distribution, CMIN and CMAX, are input by the user in routine PARAMETERS. The event is repeatedly re-scheduled throughout the simulation unless all weapons in the company sustain damage and can no longer use the ammunition.

Events CO.RESUPPLY.ARR, REDISTRIBUTE, and FIREKILL call routine COM.AMMO for the company receiving resupply in order to obtain an immediate update of the current situation prior to executing their respective program segments. These calls simulate a company commander checking the unit's resources prior to any resupply action.

Figure 5 gives an example of how a company LON is calculated for APDS ammunition. In this example the

first platoon data is taken from the platoon LON example given in Figure 4. The company example depicts 2nd platoon having 4 M1 tanks, each tank with a stowed load of 40 AP rounds and total platoon on-hand assets of 80 AP rounds.

PLT	SYSTEM	WPN	AMMO	STOWED LOAD	ON-HAND
1	tank	M1	AP	80	60
2	tank	M1	AP	160	80
3	tank	M1	AP	80	20

COMPANY LON THRESHOLDS (AP)

"5" \geq .85

"4" \geq .60

"3" \geq .40

"2" \geq .20

"1" \geq 0.0

Percent = Tot Co. on-hand Ammo / Sum of Plt stowed loads
 = 160 / 320
 = .5

Therefore since .60 \geq .50 \geq .40

COMLON = "3"

Figure 5: Company LON Example

In a similar manner it can be seen that the 3rd platoon has 2 M1 tanks remaining with 20 AP rounds between them and a stowed load of 80 rounds, 20 per tank.

3. Requests for Resupply

Resupply requests are made by a company and sent to the battalion's S-4 based on the information passed up the chain of command from the individual weapon systems through the subordinate platoons. Within the model logic, the periodic updating of LON information up to company level is the key to transmission of these resupply needs. Beyond the company level a requisition processing system replaces the LON concept. Prior to the submission of a "formal" requisition by the company to the S-4, several informal actions take place which key the submission of a requisition for a particular ammunition type. These actions occur at the weapon, platoon, and company levels. This section explains this informal process which results in the Battalion S-4 receiving a valid requisition from the company.

a. Weapon Systems

Weapon systems do not request ammunition resupply; they simply pass their most current knowledge to

the platoon at the time a platoon check is made. In the event that a weapon system exhausts its supply of an ammunition type, an immediate scheduling of the event UP.PLT.AMMO is made. This logic simulates a weapon system informing its platoon leader of the situation, and the platoon leader, in turn, performing a quick check of the rest of the platoon to see how extensive the problem is.

b. Platoon

Platoons do not request resupply; they periodically check all subordinate systems and pass on information for each ammunition type to the company when the company checks on its ammunition status. In the event that a platoon exhausts its supply of an ammunition type, an immediate scheduling of the event UP.COM.AMMO is made. This logic simulates a platoon leader informing the unit's company commander of the situation, and the company commander, in turn, performing a quick check of the rest of the company to see how extensive the problem is.

c. Company

The company is the first level of command permitted to create and submit resupply requisitions in order to correct deficiencies in a unit's ammunition

posture. Within routine CO.AMMO, changes in the Level of Need computed for an ammunition type trigger the creation of a resupply request, RES.REQ, for that specific ammunition type. These requests are transmitted to battalion supply by scheduling the event UP.S4.AMMO and passing the company's requisition list as an argument to battalion. The scheduled time of arrival for the request is determined by drawing a random number from a uniform distribution. The delimiting times for use in the distribution, MINTRIP and MAXTRIP, are input in routine PARAMETERS. Use of this distribution to schedule the event time models the delay caused by the necessity to physically carry the requests to the supply point.

d. Battalion

For the purposes of this model, a battalion does not submit requests for resupply. Convoys returning from a company resupply mission are assumed to have been reloaded at the ASP/ATP with exactly the same quantity of ammunition they had just delivered to a company. In this way, battalion stocks are constantly refilled.

4. Supply Response - Action by the S-4

Storage and issue of a battalion's reserve ammunition is simulated in event UP.S4.AMMO. The purpose of UP.S4.AMMO is to model the actions of a battalion S-4 officer allocating his limited ammunition and transportation assets in response to requests from the supported units. The event is scheduled in routine COM.AMMO when a resupply request is created. Functionally, the routine accomplishes these actions through evaluation of the following considerations: the availability of supply and transportation assets; the need to maximize the use of shipping space on board resupply vehicles; the need to adjust shipments in the face of priority requests; and control of the dispatch of resupply convoys. The subsections which follow discuss each of these considerations. Significantly, in the program logic as it exists, resupply convoys are limited to one stop deliveries. Multiple unit deliveries are not permitted.

a. Availability of Supply and Transportation Assets

Stock accountability of ammunition is maintained for each CLASS V (ammunition) item belonging to a supply officer in a temporary entity SCL.V.ITEM. Resupply requests

to the S-4 are matched against the on-hand balance in this temporary entity to determine if the supplies are available for release. A concurrent determination of the availability of transportation assets is made through a comparison of the total weight and cube available on resupply vehicles for shipping and the total weight and cube requested.

b. Maximization of Shipping Space

Program logic allows more than one type of ammunition to be loaded on a single truck. This models the S-4 seeking to effectively utilize the limited resources at his disposal. The identity of ammunition stocks released to fill a request is modelled through the creation of a temporary entity, T.CGO. This level of detail permits a determination of the total cargo manifest loaded on any individual truck.

c. Adjustments Due to Priority Requisitions

A key assumption modelled in the program is that a unit commander will order the quantity of supplies necessary to refill the unit's base load for an ammunition type. As such, in the face of multiple priority 1 and priority 2 requisitions and limited transportation assets, program logic models an S-4 decision to reduce fill on

individual requisitions in order to permit a greater number of requisitions to be filled. This reduction in fill is flexible subject only to the maximum lower bounds C.L1.PCT (CRITICAL LON 1 PERCENT) and C.L2.PCT (CRITICAL LON 2 PERCENT). These delimiters are input in routine PARAMETERS.

E. RESUPPLY ACTIVITIES - RECEIPT OF SUPPLIES

The transport of supplies to fill requisitions basically follows the reverse path of the requisition flow. This is to say that supplies are issued through the chain of command back to the weapon systems. At each level of command, new decisions are made as to the apportionment of the supplies to subordinate levels based on the most current information available. In executing the applicable routines modelling this process the first step performed is an update of the ammunition status of all levels. The apportionment process itself involves the repetitive computation at each level of a ratio (subordinate need / total unit need) times the quantity delivered. The pattern for this process is set in event CO.RESUPPLY.ARRIVE. This event is scheduled to mark the arrival of a resupply convoy at a company unit. Additionally, two other instances involving a similar reapportionment of ammunition were included in the model.

These are FIREKILL and REDISTRIBUTE. Event FIREKILL models a platoon leader's decision to distribute the ammunition assets from non-functional weapon systems to the remainder of the platoon. Event REDISTRIBUTE models an assumed standard operating procedure that requires platoons to cross-level ammunition assets between weapons after a unit move is executed. This routine differs from the two previous in that the basis for distribution of the assets is the ratio of the weapon system's stowed load to the platoon's base load times the total quantity required for the platoon.

APPENDIX B

PROGRAM DOCUMENTATION

This appendix provides a detailed explanation of the code developed for the ammunition resupply model. For this discussion, the model has been broken out into its major routines and events with each being discussed separately. The PREAMBLE section contains a detailed definition of every entity, attribute, set, and global variable used in the program. Thereafter, discussion of routines and events includes: an abbreviated glossary of terms, a listing of the program code, and a line by line description of the code. All definitions within a section are grouped by their SIMSCRIPT category then listed alphabetically. If an abbreviation is unclear an unabbreviated name is given in parentheses beside it.

A. "PREAMBLE"

The preamble provides the compiler with definitions regarding: entities, attributes, and sets; events and routines; global variables and arrays. Many of the descriptors used in this preamble are taken directly from

the current STAR model. Those taken directly from STAR are redefined here for clarity purposes and can be identified by an asterisk(*).

1. Routines

The routines of this model are described in detail in sections C through N of this appendix. The routines used are as follows:

BLU.CREATE	PARAMETERS
BASIC.LOAD	W.AMMO
P.CLASS.V	COM.AMMO
BATTLE	UP.DATE
FILE.UP.DATE	LOAD.THE.TRUCKS
WT.AND.CUBE	PRI.RESUPPLY

2. Events

The events for this model are explained in detail in sections N through Z of this appendix. The events of the model are:

B.UP.DATE	BAT.L.TIME
FIREKILL	BN.ARRIVE
CO.RESUPPLY.AR	MOVE
UP.S4.AMMO	REDISTRIBUTE
STOP.SIMULATION	UP.COM.AMMO
UP.PLT.AMMO	UP.W.AMMO

3. Entities

The entities of any SIMSCRIPT program are either permanent, meaning they remain active throughout the program's execution, or temporary, meaning they can be created or destroyed during program execution. Definitions of both type of entities are listed below.

Permanent Entities

COMPANY.COMMANDER(*). Used to model the company commander's thought process. Owns sets containing the unit's platoons(CO.UNIT composed of PLATOON.LEADERS) and the unit's unique ammunition listing (CO.AMMO composed of CCL.V.ITEMS).

PLATOON.LEADER(*). Used to model the platoon leader's thought process. Owns sets containing the unit's weapon systems(PLAT.UNIT composed of TANKS) and the unit's ammunition listing (PLT.AMMO composed of PCL.V.ITEMS).

SUPPLY.OFFICER. Used to model a battalion supply officer's thought process. Owns the following sets:

S.AMMO(SUPPLY AMMUNITION). Contains the various ammunition types (SCL.V.ITEMS) which each supply officer must stock.

S.UNIT(SUPPLY UNIT). Contains the unit's supply vehicles(TANKS).

SWANT.LIST(SUPPLY WANT LIST). Contains the unit's outstanding requisitions(RES.REQ) that must be filled.

SCONVOY(SUPPLY CONVOY). Set of ELEMENTS which make up a convoy. ELEMENTS, in turn, is composed of a set of supply vehicles(TANKS) designated for a supply mission.

Temporary Entities

CCL.V.ITEM(COMPANY CLASS V ITEM). Holds information for a company about a particular ammo type owned by the unit. Belongs to the set C.AMMO.

CONVOY. Holds information as to the type and amount of supplies being sent to a particular unit. Owns ELEMENTS which make up a convoy. ELEMENTS, in turn, owns the trucks(TANKS) that have been designated to carry the supplies.

PCL.V.ITEM(PLATOON CLASS V ITEM). Holds information for a platoon about a particular ammo type used by the unit. Belongs to the set PLT.AMMO.

RES.REQ (RESUPPLY.REQUISITION). Models a present day requisition form. Provides information between users and the supply system. A RES.REQ is used to hold requirements information for a variety of purposes through its membership in various sets. These are:

C.WANT.LIST. Company information owned by a COMPANY.COMMANDER.

SWANT.LIST. Supply information owned by a SUPPLY.OFFICER.

C.CGO.LIST. Convoy cargo list owned by a CONVOY.

SCL.V.ITEM (SUPPLY CLASS V ITEM). Holds information for a supply unit about a particular ammo type used by the unit. Belongs to the set S.AMMO.

T.CGO(TRUCK CARGO). Holds information concerning the supplies loaded on a truck. Belongs to the set CARGO.

TANK(*). Represents any vehicle or weapon system on the battlefield. Used to distinguish individual vehicles as to type and function. Tanks belong to several distinguishing sets:

TNK.ALIVE (TANK ALIVE) (*). Owned by the system this set keeps track of the alive/dead status of individual TANKs.

PLAT.UNIT(PLATOON UNIT). Combat systems and vehicles belong. Owned by a PLATOON.LEADER.

S.UNIT(SUPPLY UNIT). Supply vehicles only belong to this set which is owned by a SUPPLY.OFFICER.

M.ELEMENTS. Specifies membership in a CONVOY.

4. Attributes

Permanent Attributes (INTEGER)

N.CCL.V.ITEMS (NUMBER OF COMPANY CLASS V ITEMS).

COMPANY.COMMANDER attribute specifying the number of ammunition types(CCL.V.ITEMS) used by the company.

N.PCL.V.ITEMS (NUMBER OF PLATOON CLASS V ITEMS).

PLATOON.LEADER attribute specifying the number of ammunition types(PCL.V.ITEMS) used by the platoon.

N.SCL.V.ITEMS (NUMBER OF SUPPLY CLASS V ITEMS).

SUPPLY.OFFICER attribute specifying the number of ammunition types(SCL.V.ITEMS) used by the battalion supply officer.

PCO.CDR(PLATOON COMPANY COMMANDER). Specifies a platoon's commander.

REQN(REQUISITION). Attribute of a COMPANY.COMMANDER specifying the total number of resupply requests filed by a commander.

SCO.CDR(SUPPLY COMPANY COMMANDER). Specifies a supply unit's commander.

S4.OFF(S4 OFFICER). Attribute of a COMPANY.COMMANDER specifying the unit's supply officer.

Temporary_Attributes_(ALPHA)

STATUS. Attribute of a RES.REQ indicating where the request currently is. Its possible values are: TOS4, TOCO, and TOATP.

CNOMEN (COMPANY NOMENCLATURE). Attribute of a CCL.V.ITEM containing the nomenclature of a particular ammo.

PNOMEN (PLATOON NOMENCLATURE).

RNOMEN (RESUPPLY NOMENCLATURE). Attribute of a RES.REQ specifying the requested ammo's nomenclature.

SNOMEN (SUPPLY NOMENCLATURE). Attribute of a SCL.V.ITEM specifying the requested ammo's nomenclature.

TNOMEN (T.CGO NOMENCLATURE). Attribute of T.CGO containing the name of the ammunition item carried.

Temporary_Attributes_(INTEGER)

AMMO1(*) (AMMUNITION 1). This variable is used as a shortened form for AP.TOW ammunition.

AMMO2(*) (AMMUNITION 2). This variable is used as a shortened form for HE.DRAG ammunition.

AMMO3(*) (AMMUNITION 3). This variable is used as a shortened form for AW1.OR.MSL3 ammo.

AMMO4(*) (AMMUNITION 4). This variable is used as a shortened form for AW2.OR.ADM ammo.

AMMO5(*) (AMMUNITION 5). Actual on-hand balance of the fifth ammo type fired by a TANK.

AMMO6(*) (AMMUNITION 6). Actual on-hand balance of the sixth ammo type fired by a TANK.

AP.TOW(*) (ARMOR PIERCING/TOW) Actual on-hand balance of the first ammo type fired by a TANK.

TAC(TANK AMMUNITION CODE). Supply code which points to a specific ammunition fired by a TANK. Six are specified on a TANK:

TAC1(TANK AMMUNITION CODE 1). Contains the code value for the first ammo type fired by a TANK.

TAC2(TANK AMMUNITION CODE 2). Contains the code value for the second ammo type fired by a TANK.

TAC3(TANK AMMUNITION CODE 3). Contains the code value for the third ammo fired by a TANK.

TAC4(TANK AMMUNITION CODE 4). Contains the code value for the fourth ammo fired by a TANK.

TAC5 (TANK AMMUNITION CODE 5). Contains the code value for the fifth ammo fired by a TANK.

TAC6 (TANK AMMUNITION CODE 6). Contains the code value for the sixth ammo fired by a TANK.

AW1.OR.MSL3(*) (ALTER WEAPON1 OR MISSILE3 AMMUNITION). Actual on-hand balance for the third ammo fired by a TANK.

AW2.OR.ADM(*) (ALTER WEAPON2 OR AIR DEF MISSILE). Actual on-hand balance for the fourth ammo fired by a TANK.

C.CMBT.LOSS (COMPANY COMBAT LOSS). Attribute of a CCL.V.ITEM indicating whether the need for an ammo type is still viable.

C.MV.STATE (CONVOY MOVEMENT STATE). Indicates if a convoy has left its start point. Equals "0" at the start point and "1" if departed.

C.NUM.REQ (COMPANY NUMBER OF REQUESTS). Attribute of a CCL.V.ITEM containing the total number of requests made for that ammo type.

C.RND.CNTR (COMPANY ROUND COUNTER). Argument for the event UP.COM.AMMO, points to the weapon system it is updating.

C.SHORT (COMPANY SHORTAGE). Attribute of a CCL.V.ITEM holding the number of rounds the company is short for that round type.

CAC(COMPANY AMMUNITION CODE). Attribute of a CCL.V.ITEM
which points to a specific ammo type fired by the
company weapon systems.

CAMMO.LON(COMPANY AMMUNITION LEVEL OF NEED). Attribute of a
CCL.V.ITEM indexing the company's overall need for an
ammo type.

CCURR.LOAD(COMPANY CURRENT LOAD). Attribute of a CCL.V.ITEM
holding the company commander's knowledge of the on-hand
balance for rounds of a particular type.

CO.B.LOAD(COMPANY BASE LOAD). Attribute of a CCL.V.ITEM
holding the total number of rounds the company needs to
be at optimal fill.

CO.CNVY(COMPANY CONVOY). Argument of event CO.RESUPPLY.ARR
(COMPANY RESUPPLY ARRIVE) pointing to the CONVOY
arriving.

COCDR(COMPANY COMMANDER). Attribute of a TANK pointing to
its COMPANY.COMMANDER.

COLOR(*). Attribute of TANK indicating the TANK's force
membership.

"0" indicates RED FORCE

"1" indicates BLUE FORCE

CONTRKS(CONVOY TRUCKS). Attribute of a CONVOY specifying the
number of vehicles in a particular convoy.

CPNTR (CONVOY POINTER) . Attribute of a T.CGO pointing to the
convoy the cargo is loaded on.

CRESUPPLY.REQ (COMPANY RESUPPLY REQUEST) . Attribute of a
CCL.V.ITEM indicating whether a RES.REQ has been
submitted previously.

CU.PKG (CUBE PACKAGE) . Attribute of a SCL.V.ITEM specifying
the cube of an ammunition pallet.

DEMAND. Attribute of a SCL.V.ITEM holding the total demand
for an ammo type.

DISTR (DISTRIBUTOR) . Argument for event REDISTRIBUTE pointing
to the unit's PLATOON.LEADER

FKILL (FIREPOWER KILL) (*). Indicates whether a TANK has
sustained a firepower kill during the battle.

"0" indicates no

"1" indicates yes

MARCH.ORDER. Argument for the event MOVE holding the pointer
of the company receiving orders to move.

HE.DRAG (HIGH EXPLOSIVE/Dragon AMMUNITION) (*). Actual on-hand
balance of the second ammo type fired by a TANK.

ISSUER. Argument of event UP.S4.AMMO pointing to the S-4
currently updating.

ISSUEE. Argument of event UP.S4.AMMO pointing to the company
initiating the request for resupply.

KKILL (CATASTROPHIC KILL) (*). Indicates whether a TANK has sustained a catastrophic kill during the battle.

"0" indicates no

"1" indicates yes

MANIFEST. Attribute of a RES.REQ pointing to the CONVOY supplies are loaded on.

MAX.CUBE. Attribute of a TANK indicating its max cargo cube.

MAX.WT. Attribute of a TANK indicating its max cargo weight.

MFKILL (MOBILITY/FIREPOWER KILL) (*). Indicates whether a TANK has sustained mobility and firepower damage.

"0" indicates no

"1" indicates yes

MKILL (MOBILITY KILL) (*). Indicates whether a TANK has sustained mobility damage.

"0" indicates no

"1" indicates yes

N.T.ALLOC (NUMBER OF TRUCKS ALLOCATED). Attribute of a RES.REQ indicating the total number of trucks allocated to move a RES.REQ.

NAME (*). Indicates the number of a TANK in the battle.

ONHAND. Attribute of a SCL.V.ITEM holding the balance on-hand of stocks for an ammo type.

OH1 (ON-HAND 1). Current balance of ammunition 1 on a TANK.

OH2 (ON-HAND 2). Current balance of ammunition 2 on a TANK.

OH3 (ON-HAND 3). Current balance of ammunition 3 on a TANK.

OH4 (ON-HAND 4). Current balance of ammunition 4 on a TANK.

OH5 (ON-HAND 5). Current balance of ammunition 5 on a TANK.

OH6 (ON-HAND 6). Current balance of ammunition 6 on a TANK.

P.CMBT.LOSS (PLATOON COMBAT LOSS). Attribute of a PCL.V.ITEM indicating whether the need for an ammo type is still viable.

P.RND.CNTR (PLATOON ROUND COUNTER). Argument of event UP.PLT.AMMO pointing to the platoon currently updating.

P.SHORT (PLATOON SHORTAGE). Attribute of a PCL.V.ITEM holding the quantity of that ammo the platoon is currently short.

PAC (PLATOON AMMUNITION CODE). Attribute of a PCL.V.ITEM which points to a specific ammo type fired by the platoon.

PAMMO.LON (PLATOON AMMUNITION LEVEL OF NEED). Attribute of a PCL.V.ITEM indexing the platoon's overall need for an ammo type.

PCURR.LOAD (PLATOON CURRENT LOAD). Attribute of a PCL.V.ITEM holding the platoon leader's knowledge of the on-hand balance for rounds of a particular type.

PNOMEN (PLATOON NOMENCLATURE). Attribute of a PCL.V.ITEM containing the nomenclature of a particular ammo.

PL.B.LOAD(PLATOON BASE LOAD). Attribute of a PCL.V.ITEM
holding the total number of rounds the platoon needs to
be at optimal fill for that type ammunition.

PLTLDR(PLATOON LEADER)(*). Attribute of a TANK.

POINTER(*). Contains the machine address of a particular
TANK.

RAC(RESUPPLY AMMUNITION CODE). Attribute of an SCL.V.ITEM
which points to a specific ammo type carried by the
supply unit.

RCNVY(RESUPPLY CONVOY). Argument of event BN.ARRIVE pointing
to a specific convoy.

RDS.PKG(ROUNDS PACKAGE). Attribute of a SCL.V.ITEM
specifying the number of rounds on an ammo pallet.

REQUESTOR. Attribute of a RES.REQ pointing to the requesting
unit.

RFILL(RESUPPLY FILL). Attribute of a RES.REQ specifying the
amount of ammunition released to fill a request.

RND.CNTR(ROUND COUNTER). Argument of event UP.W.AMMO
pointing to the TANK currently updating.

RP(RELEASE POINT). Attribute of a CONVOY specifying its
destination.

RPNTR(REQUEST POINTER). Attribute of a RES.REQ containing its pointer value.

RRPNTR(RES.REQ POINTER). Attribute of T.CGO which points to the RES.REQ the supplies are meant to fill.

RQTY(RESUPPLY QUANTITY). Attribute of a RES.REQ holding the total quantity requested by a unit.

RAC(RESUPPLY AMMUNITION CODE). Attribute of a RES.REQ which points to a specific ammo being requested.

SAC(SUPPLY AMMUNITION CODE). Attribute of a SCL.V.ITEM pointing to the particular ammo carried by the supply unit.

SCREEN. Attribute of a RES.REQ indicating whether a RES.REQ has been looked at during an S-4 update.

SLOAD1(STOWED LOAD 1). Attribute of a TANK specifying the optimal load for ammo type 1.

SLOAD2(STOWED LOAD 2). Attribute of a TANK specifying the optimal load for ammo type 2.

SLOAD3(STOWED LOAD 3). Attribute of a TANK specifying the optimal load for ammo type 3.

SLOAD4(STOWED LOAD 4). Attribute of a TANK specifying the optimal load for ammo type 4.

SLOAD5 (STOWED LOAD 5) . Attribute of a TANK specifying the optimal load for ammo type 5.

SLOAD6 (STOWED LOAD 6) . Attribute of a TANK specifying the optimal load for ammo type 6.

SP (START POINT) . Attribute of a CONVOY specifying its origin.

SPACE. Attribute of a CONVOY holding the amount of loading space available on trucks in the convoy.

SPRIORITY (SUPPLY PRIORITY) . Attribute of a RES.REQ specifying the urgency of need for the ammo request.

SUPOFF (SUPPLY OFFICER) (*). Attribute of a resupply vehicle.

SYS.TYPE (SYSTEM TYPE) (*). Attribute of a TANK specifying the general system type of the entity.

1	TANK
2	Mounted Infantry
3	Dismounted Infantry
4	Artillery
5	Air
6	Air Defense
7	Supply
8	Comm/EW/Acq/Intel
9	Other

TCU (TRUCK CUBE) . Attribute of a TANK holding the maximum cube loaded on a truck.

TPNTR (TRUCK POINTER) . Attribute of a T.CGO pointing to the truck it is loaded on.

TQTY (T.CGO QUANTITY) . Attribute of a T.CGO containing the quantity loaded as cargo.

TRAC(T.CGO RESUPPLY AMMO CODE). Attribute of T.CGO pointing to the ammo type loaded as cargo.

TWT(TRUCK WEIGHT). Attribute of a TANK holding the maximum weight it can carry.

TIME. Attribute of a RES.REQ containing the time a request is initiated at the company.

WLON1(WEAPON LEVEL OF NEED 1). Weapon system urgency of need AMMO1.

WLON2(WEAPON LEVEL OF NEED 2). Weapon system urgency of need AMMO2.

WLON3(WEAPON LEVEL OF NEED 3). Weapon system urgency of need AMMO3.

WLON4(WEAPON LEVEL OF NEED 4). Weapon system urgency of need AMMO4.

WLON5(WEAPON LEVEL OF NEED 5). Weapon system urgency of need AMMO5.

WLON6(WEAPON LEVEL OF NEED 6). Weapon system urgency of need AMMO6.

WPN.TYPE(WEAPON TYPE)(*). Attribute of a TANK specifying the specific weapon system within a SYS.TYPE(SYSTEM TYPE).

WT.PKG(WEIGHT PACKAGE). Attribute of a SCL.V.ITEM specifying the weight of a pallet of the ammo being considered.

5. Sets

The sets used in the model are listed and defined as follows:

C.CGO.LIST (CONVOY CARGO LIST). Owned by a CONVOY. Members are RES.REQs.

CO.AMMO (COMPANY AMMUNITION). Owned by a COMPANY.COMMANDER. Members are the unit's CCL.V.ITEMS.

CO.UNIT (COMPANY UNIT). Owned by a COMPANY.COMMANDER. Members are unit PLATOON.LEADERS.

CARGO (TRUCK CARGO). Owned by a supply vehicle. Members are the supplies (T.CGO) loaded.

CWANT.LIST (COMPANY WANT LIST). Owned by a COMPANY.COMMANDER. Attributes are the unit's outstanding resupply requests.

ELEMENTS. Owned by a convoy. Members are TANKs in the convoy.

PLAT.UNIT (PLATOON UNIT). Owned by a PLATOON.LEADER. Members are unit combat vehicles.

PLT.AMMO (PLATOON AMMUNITION). Owned by a PLATOON.LEADER. Members are the platoon PCL.V.ITEMS.

S.AMMO (SUPPLY AMMUNITION). Owned by a SUPPLY.OFFICER. Members are the supply unit's SCL.V.ITEMS.

S.UNIT(SUPPLY UNIT). Owned by a SUPPLY.OFFICER. Members are unit supply trucks.

SCONVOY(SUPPLY CONVOY). Owned by a SUPPLY.OFFICER. Members are convoys dispatched by the supply unit.

SREQN.LIST(SUPPLY REQUISITION LIST). Owned by a SUPPLY.OFFICER. Members are the supply unit's outstanding RES.REQS sent to the ATP.

SUP.OFF(SUPPLY OFFICER). Attribute of TANK pointing to its supply officer.

SWANT.LIST(SUPPLY WANT LIST). Owned by a SUPPLY.OFFICER. Members are RES.REQS from the combat units.

TNK.ALIVE(TANK ALIVE) (*). Owned by the system. Members are vehicles still alive within the simulation.

6. Global Variables

The global variables used in the model are either alpha, integer or real. An explanation of their use is as follows:

Global Variables_(ALPHA)

NOMEN(NOMENCLATURE). Specifies the names of the rounds played.

Global Variables_(INTEGER)

CCODE(COMPANY CODE). Holds the pointer value for a company's
CCL.V.ITEMs

CNUM(COMPANY NUMBER) (*). Specifies the number of
COMPANY.COMMANDERs created.

CSTREAM(COMPANY STREAM). Specifies the random number stream
used for company calculations.

N.SYS(NUMBER OF SYSTEMS). Specifies the number of weapon
systems created for the simulation.

N.TANKS(NUMBER OF TANKS). Specifies the number of vehicles
created for the simulation.

N.WPN.TYPES(NUMBER OF WEAPON TYPES). Specifies the number of
weapons created for the simulation.

PCODE(PLATOON CODE) (2-d). Holds the pointer value for a
platoon's PCL.V.ITEMs.

PNUM(PLATOON NUMBER) (*). Specifies the number of
PLATOON.LEADERS created for a simulation.

PSTREAM(PLATOON STREAM). Specifies the random number stream
to be used for platoon calculations.

RCODE(REQUEST CODE) (2-d). Holds the pointer value for
resupply requests created.

RSTREAM(RESUPPLY STREAM). Specifies the random number stream
to be used for resupply calculations.

SCODE(SUPPLY CODE) (2-d). Holds the pointer value for a supply unit's SCL.V.ITEMs.

SNUM(SUPPLY NUMBER). Specifies the number of SUPPLY.OFFICERS created for a simulation.

TSTREAM(TRIP STREAM). Specifies the random number stream to be used for movement calculations.

WSTREAM(WEAPON STREAM). Specifies the random number stream to be used for weapon update calculations.

Global Variables (REAL)

B.END(BATTLE END). Holds the termination time for a day's battle.

B.START(BATTLE START). Holds the beginning time for a day's battle.

C.L1PCT(CRITICAL LON1 PERCENTAGE). Holds the maximum percentage that LON1 requisitions may be reduced to in order to release space on a convoy for other critical LON1 and LON2 requests.

C.L2CU(CRITICAL LON2 CUBE). Holds the maximum percent of cube that LON2 requisitions may be cut to in order to release space on a convoy for other critical LON1 and LON2 requisitions.

C.L2WT (CRITICAL LON2 WEIGHT) . Holds the maximum percent of weight that LON2 requisitions may be cut to in order to release space on a convoy for other critical LON1 and LON2 requisitions.

CMAX (COMPANY MAXIMUM) . The maximum time that can elapse before a company will update its ammunition status.

CMIN (COMPANY MINIMUM) . The minimum time that can elapse before a company will update its ammunition status.

COMLON (COMPANY LEVEL OF NEED) (2-d) . Array holding the value of a COMPANY.COMMANDER's cutoff percentages for the five levels of need which can be assigned. Dimensioned as MAX.CL.V.ITEMS (MAX CLASS V ITEMS) by 5.

MAXTRIP (MAX TRIP) . The maximum time required for a convoy to reach its intended destination.

MINTRIP (MIN TRIP) . The minimum time required for a convoy to reach its intended destination.

PLTLON (PLATOON LEVEL OF NEED) (2-d) . Array holding the value of a PLATOON.LEADERS's cutoff percentages for the five levels of need which can be assigned. Dimensioned as MAX.CL.V.ITEMS (MAX CLASS V ITEMS) by 5.

PMAX (PLATOON MAXIMUM) . The maximum time that can pass before a platoon will update its ammunition status.

PMIN(PLATOON MINIMUM). The minimum time that can pass before a platoon will update its ammunition status.

POD(PROBABILITY OF DAMAGE) (2-d). Array holding the probability of damage for the various weapon types and types of damage. Dimensioned N.WPN.TYPES(NUMBER OF WEAPON TYPES) by 4.

POF(PROBABILITY OF FIRE) (2-d). Array holding the probability of firing for the various weapon types and ammunitions fired. Dimensioned N.WPN.TYPES(NUMBER OF WEAPON TYPES) by 4.

ROF(RATE OF FIRE) (2-d). Specifies the rates of fire for the six weapons of any weapon system. Dimensioned N.WPN.TYPES(NUMBER OF WEAPON TYPES) by 6.

WMAX(WEAPON MAXIMUM). The maximum time that can pass before a weapon will update its ammunition status.

WMIN(WEAPON MINIMUM). The minimum time that can pass before a weapon will update its ammunition status.

WPNLON(WEAPON LEVEL OF NEED) (2-d). Array holding the value of a weapon system's cutoff percentages for the five levels of need which can be assigned. Dimensioned as MAX.CL.V.ITEMS(MAX CLASS V ITEMS) by 5.

7. Listing

```

1  PREAMBLE
2  DEFINE WPNLON,PLTLON,AND COMLON AS REAL,2-DIM ARRAYS
3  DEFINE ROF AS INTEGER,2-DIM ARRAY
4  DEFINE NOMEN AS AN ALPHA,1-DIM ARRAY
5  DEFINE PLACES AS A 2-DIM INTEGER ARRAY
6  ''
7  DEFINE POD AS REAL,2-DIM ARRAY
8  GENERATE LIST ROUTINES
9  ''
10 THE SYSTEM OWNS SOME TNK.ALIVE
11 ''
12 ''
13 PERMANENT ENTITIES
14 ''
15 EVERY COMPANY.COMMANDER HAS AN N.CCL.V.ITEMS,
16 AN S4.OFF,A REQN,
17 OWNS A CO.UNIT,A CWANT.LIST,AND SOME CO.AMMO
18 DEFINE N.CCL.V.ITEMS,S4.OFF,REQN AS INTEGER
19 VARIABLES
20 ''
21 EVERY PLATOON.LEADER HAS A PCO.CDR,AN N.PCL.V.ITEMS,
22 MAY BELONG TO A CO.UNIT,
23 AND MAY OWN A PLAT.UNIT,AND A PLT.AMMO
24 DEFINE PCO.CDR, N.PCL.V.ITEMS AS INTEGER VARIABLES
25 ''
26 EVERY SUPPLY.OFFICER HAS A SCO.CDR,A N.SCL.V.ITEMS,
27 OWNS SOME S.AMMO, AN S.UNIT,AN SWANT.LIST,
28 AN SREQN.LIST, AND AN SCONVOY
29 DEFINE SCO.CDR, N.SCL.V.ITEMS AS INTEGER VARIABLES
30 ''
31 ''
32 TEMPORARY ENTITIES
33 ''
34 EVERY TANK HAS A NAME,A COLOR,A SYS.TYPE,A WPN.TYPE,
35 A POINTER,AN AP.TOW,AN HE.DRAG,AN AW1.OR.MSL3,
36 AN AW2.OR.ADM,AN AMMO5,AN AMMO6,
37 A SLOAD1,A SLOAD2,A SLOAD3,A SLOAD4,A SLOAD5,
38 A SLOAD6,AN OH1,AN OH2,AN OH3,AN OH4,AN OH5,AN OH6,
39 A WLON1,A WLON2,A WLON3,A WLON4,A WLON5,A WLON6,
40 AN TAC1,AN TAC2,AN TAC3,AN TAC4,AN TAC5,AN TAC6,
41 A PLTLDR,A COCDR,A SUPOFF,A MAX.WT,A MAX.CUBE,
42 A TWT,A TCU
43 AN MKILL,AN FKILL,AN MFKILL,A KILL,
44 AND MAY BELONG TO A TNK.ALIVE,A PLAT.UNIT,A S.UNIT,
45 AND A ELEMENTS,AND MAY OWN SOME CARGO
46 ''
47 DEFINE N.TANKS AS AN INTEGER VARIABLE
48 DEFINE NAME,COLOR,SYS.TYPE,WPN.TYPE,SUP.OFF,POINTER,
49 AP.TOW,HE.DRAG,AW1.OR.MSL3,AW2.OR.ADM,AMMO5,AMMO6,
50 SLOAD1,SLOAD2,SLOAD3,SLOAD4,SLOAD5,SLOAD6,
51 OH1,OH2,OH3,OH4,OH5,OH6,
52 WLON1,WLON2,WLON3,WLON4,WLON5,WLON6,
53 TAC1,TAC2,TAC3,TAC4,TAC5,TAC6,
54 PLTLDR,COCDR,SUPOFF,MAX.WT,MAX.CUBE,
55 TWT,TCU
56 MKILL,FKILL,MFKILL,AND KILL AS INTEGER VARIABLES
57 ''
58 EVERY T.CGO HAS A TPNTR,A RRPNTR, A TRAC, A TNOMEN,
59 A TOTY,AND MAY BELONG TO A CARGO
60 DEFINE TPNTR,RRPNTR,TRAC,TOTY AS INTEGER VARIABLES
61 DEFINE TNOMEN AS AN ALPHA VARIABLE
62 ''
63 EVERY CONVOY HAS AN SP,AN RP,A CONTRKS,A SPACE,
64 A C.MV.STATE,A CPNTR,MAY BELONG TO A SCONVOY,MAY OWN

```



```

65 SOME ELEMENTS, AND A C.CGO.LIST
66 DEFINE CPNTR, SP, RP, SPACE, C.MV.STATE, AND CONTRKS AS
67 INTEGER VARIABLES
68 ''
69 EVERY PCL.V.ITEM HAS A PAC, A PNOMEN, A PL.B.LOAD,
70 A PCURR.LOAD, A PAMMO.LON, A P.CMBT.LOSS, A P.SHORT, AND
71 MAY BELONG TO A PLT.AMMO
72 DEFINE PAC, PL.B.LOAD, PCURR.LOAD,
73 PAMMO.LON, P.SHORT, P.CMBT.LOSS AS INTEGER VARIABLES
74 DEFINE PNOMEN AS AN ALPHA VARIABLE
75 ''
76 EVERY CCL.V.ITEM HAS A CAC, A CNOMEN, A CO.B.LOAD,
77 A CCURR.LOAD, A C.NUM.REQN, A CAMMO.LON, A C.SHORT,
78 A C.CMBT.LOSS, AND MAY BELONG TO A CO.AMMO
79 DEFINE CAC, CO.B.LOAD, CCURR.LOAD, C.NUM.REQN, C.SHORT,
80 C.CMBT.LOSS, CAMMO.LON AS INTEGER VARIABLES
81 DEFINE CNOMEN AS AN ALPHA VARIABLE
82 ''
83 EVERY SCL.V.ITEM HAS AN SAC, AN SNOMEN, A WT.PKG,
84 A CU.PKG, A RDS.PKG, A DEMAND, AN ONHAND, AND
85 BELONGS TO AN S.AMMO
86 DEFINE SAC, RDS.PKG, DEMAND, ONHAND AS
87 INTEGER VARIABLES
88 DEFINE SNOMEN AS AN ALPHA VARIABLE
89 DEFINE WT.PKG, AND CU.PKG AS REAL VARIABLES
90 ''
91 EVERY RES.REQ HAS A REQUESTOR, A RQTY, A RAC,
92 AN RNOMEN, AN SPRIORITY, A STATUS, A TIME, AN RPILL,
93 AN N.T.ALLOC, A MANIFEST, A SCREEN, AN RPNTR, AND MAY
94 BELONG TO A C.CGO.LIST, A CWANT.LIST,
95 AN SREQN.LIST, AND AN SWANT.LIST
96 DEFINE REQUESTOR, RQTY, RAC, SPRIORITY, MANIFEST,
97 RPNTR, SCREEN, RPILL, N.T.ALLOC AS INTEGER VARIABLES
98 DEFINE TIME AS A REAL VARIABLE
99 DEFINE STATUS AS AN ALPHA VARIABLE
100 DEFINE RNOMEN AS A ALPHA VARIABLE
101 ''
102 ''
103 ''
104 '' GENERAL DEFINITIONS
105 ''
106 DEFINE PNUM, CNUM, SNUM, WSTREAM, PSTREAM, CSTREAM,
107 TSTREAM, RSTREAM, N.SYS, AND N.WPN.TYPES
108 AS INTEGER VARIABLES
109 DEFINE WMIN, WMAX, PMIN, PMAX, CMIN, CMAX,
110 MINTRIP, AND MAXTRIP AS REAL VARIABLES
111 DEFINE B.START, AND B.END AS REAL VARIABLES
112 DEFINE SWANT.LIST AS A SET RANKED BY LOW SPRIORITY,
113 THEN BY LOW TIME
114 DEFINE PCODE AS AN INTEGER, 2-DIM ARRAY
115 DEFINE CCODE AS AN INTEGER, 2-DIM ARRAY
116 DEFINE SCODE AS AN INTEGER, 2-DIM ARRAY
117 DEFINE RCODE AS AN INTEGER, 2-DIM ARRAY
118 DEFINE AMMO1 TO MEAN AP.TOW
119 DEFINE AMMO2 TO MEAN HE.DRAG
120 DEFINE AMMO3 TO MEAN AW1.OR.MSL3
121 DEFINE AMMO4 TO MEAN AW2.OR.ADM
122 DEFINE C.L2WT, C.L2CU, C.L2PCT, C.L1PCT
123 AS REAL VARIABLES
124 ''
125 ''
126 '' EVENT NOTICES
127 ''
128 EVENT NOTICES INCLUDE STOP.SIMULATION, B.UP.DATE,
129 AND BAT.L.TIME
130 EVERY UP.W.AMMO HAS A RND.CNTR
131 EVERY UP.PLT.AMMO HAS A P.RND.CNTR
132 EVERY UP.COM.AMMO HAS A C.RND.CNTR
133 EVERY UP.S4.AMMO HAS A ISSUER, AND AN ISSUEE

```



```

134 EVERY MOVE HAS A MARCH.ORDER
135 EVERY CO.RESUPPLY.ARR HAS A CO.CNVY
136 EVERY BN.ARRIVE HAS AN RCNVY
137 EVERY REDISTRIBUTE HAS A DISTR
138 EVERY FIREKILL HAS A VICTIM
139 DEFINE VICTIM,RND.CNTR,P.RND.CNTR,C.RND.CNTR,ISSUER,
140 ISSUEE,CO.CNVY,MARCH.ORDER,RCNVY,AND DISTR
141 AS INTEGER VARIABLES

```

B. "MAIN"

The purpose of the main program is to call routines that create blue forces, to schedule the events that generate data, and to start/stop the simulation.

EVENT NOTICES

```

B.UP.DATE(BATTLE UPDATE), BAT.L.TIME(BATTLE TIME),
STOP.SIMULATION

```

RECURSIVE VARIABLES (REAL)

SIM.STOP Holds the time for simulation termination.

ROUTINES CALLED

```

BLU.CREATE

```

PROGRAM LISTING

```

1  MAIN
2  ''
3  DEFINE SIM.STOP AS A REAL VARIABLE
4  READ SIM.STOP
5  SCHEDULE A STOP.SIMULATION IN SIM.STOP DAYS
6  SKIP 2 CARDS
7  CALL BLU.CREATE
8  SCHEDULE A BAT.L.TIME NOW
9  SCHEDULE A B.UP.DATE NOW
10 PRINT 1 LINE THUS
    START SIMULATION
11 START SIMULATION
12 STOP
13 END

```

EXPLANATION OF CODE

Line 3 Defines recursive variables used in the routine.

Line 4 Reads simulation stop time.
Line 5 Schedules the event STOP.SIMULATION.
Line 7 Calls routine BLU.CREATE.
Line 8 Schedules the first BAT.L.TIME event.
Line 9 Schedules printing of the first battle summary.
Line 10 Prints a statement to mark simulation start.
Line 11 System command to start the simulation.

C. "ROUTINE BLU.CREATE"

Routine BLU.CREATE is called from the main program to create the entities of each blue unit. After creating these entities, it establishes their attributes and files each entity in its appropriate set.

EVENTS_SCHEDULED

UP.W.AMMO(UPDATE WEAPON AMMO).

GLOBAL_VARIABLES_(INTEGER)

CNUM(COMPANY NUMBER). Specifies the number of

COMPANY.COMMANDERS created.

CSTREAM(COMPANY RANDOM NUMBER STREAM).

N.COMPANY.COMMANDERS(NUMBER OF COMPANY COMMANDERS).

N.PLATOON.LEADERS(NUMBER OF PLATOON LEADERS).

N.SUPPLY.OFFICERS(NUMBER OF SUPPLY OFFICERS).

N.TANKS(NUMBER OF TANKS).

PNUM(NUMBER OF PLATOONS).

PSTREAM(PLATOON RANDOM NUMBER STREAM) .

SNUM(NUMBER OF SUPPLY OFFICERS) .

GLOBAL_VARIABLES_(REAL)

CMAX(COMPANY MAXIMUM) . The maximum time that can pass before
a company will update its ammunition status.

CMIN(COMPANY MINIMUM) . The minimum time that can pass before
a company will update its ammunition status.

PMAX(PLATOON MAXIMUM) . The maximum time that can pass before
a platoon will update its ammunition status.

PMIN(PLATOON MINIMUM) . The minimum time that can pass before
a platoon will update its ammunition status.

WMAX(WEAPON MAXIMUM) . The maximum time that can pass before
a weapon will update its ammunition status.

WMIN(WEAPON MINIMUM) . The minimum time that can pass before
a weapon will update its ammunition status.

PERMANENT_ENTITIES

COMPANY.COMMANDER, PLATOON.LEADER, SUPPLY.OFFICER

PERMANENT_ATTRIBUTES_(INTEGER)

N.CCL.V.ITEMS. Number of company class V items (unique to a
company commander) .

N.PCL.V.ITEMS. Number of platoon class V items (UNIQUE TO A
PLATOON LEADER) .

N.SCL.V.ITEMS. Number of supply class V items(UNIQUE TO A
SUPPLY OFFICER) .

PCO.CDR. (PLATOON COMPANY COMMANDER) .

SCO.CDR. (SUPPLY COMPANY COMMANDER) .

S4.OFF. (COMPANY S-4 OFFICER) .

RECURSIVE_VARIABLES_(INTEGER)

I - Loop index.

ROUTINES_CALLED

BASIC.LOAD, PARAMETERS

SETS

CO.UNIT(COMPANY UNIT) . Owned by a COMPANY.COMMANDER.

Members are the unit's PLATOON.LEADERS.

PLAT.UNIT(PLATOON UNIT) . Owned by a PLATOON.LEADER. Members
are the unit's combat vehicles(TANKs) .

S.UNIT(SUPPLY UNIT) . Owned by a SUPPLY.OFFICER. Members are
the unit's supply vehicles(TANKs) .

TNK.ALIVE(TANK ALIVE) . Owned by the system. Members are
vehicles still alive within the simulation.

TEMPORARY_ENTITIES

TANK. A temporary entity used to represent all vehicles on
the battlefield. Its attributes distinguish the
individual vehicles as to type and function.

TEMPORARY_ATTRIBUTES_(INTEGER)

AMMO5 (AMMUNITION 5) .

AMMO6 (AMMUNITION 6) .

AP.TOW (ARMOR-PIERCING/TOW) . Ammunition 1.

TAC1. (TANK AMMUNITION CODE 1) .

TAC2. (TANK AMMUNITION CODE 2) .

TAC3. (TANK AMMUNITION CODE 3) .

TAC4. (TANK AMMUNITION CODE 4) .

TAC5. (TANK AMMUNITION CODE 5) .

TAC6. (TANK AMMUNITION CODE 6) .

AW1.OR.MSL3 (ALTERNATE WEAPON 1 OR MISSILE 3) . Ammunition 3.

AW2.OR.ADM (ALTERNATE WEAPON 2 OR AIR DEFENSE MISSILE) .

Ammunition 4.

COCDR (COMPANY COMMANDER) . Of a TANK.

COLOR This attribute of a TANK indicates the TANK's force membership. "0" indicates RED FORCE, "1" indicates BLUE.

HE.DRAG (HEAT/DRAGON ROUNDS) . Ammunition 2.

MAX.CUBE. Indicates the max cargo cube a resupply vehicle(TANK) is designed to move.

MAX.WT. Indicates the max cargo weight a resupply vehicle(TANK) is designed to move.

NAME. Indicates the number of a TANK in the battle.

PLTLDR (PLATOON LEADER) . Of TANK.

POINTER. Combat vehicle's machine address.

RND.CNTR(ROUND COUNTER). Argument of routine W.AMMO(WEAPON AMMUNITION) carrying the value of the TANK updating.

SLOAD1(STOWED LOAD 1). Optimal load ammo type 1.

SLOAD2(STOWED LOAD 2). Optimal load ammo type 2.

SLOAD3(STOWED LOAD 3). Optimal load ammo type 3.

SLOAD4(STOWED LOAD 4). Optimal load ammo type 4.

SLOAD5(STOWED LOAD 5). Optimal load ammo type 5.

SLOAD6(STOWED LOAD 6). Optimal load ammo type 6.

SUPOFF(SUPPLY OFFICER). Of TANK.

SYS.TYPE(SYSTEM TYPE). Of TANK.

TWT(TRUCK WEIGHT). Maximum cargo weight for a vehicle.

TCU(TRUCK CUBE). Maximum cargo cube for a vehicle.

WPN.TYPE(WEAPON TYPE). Of TANK.

GLOBAL_VARIABLES_(INTEGER)

WSTREAM(WEAPON RANDOM NUMBER STREAM).

ARGUMENT_(INTEGER)

RND.CNTR(ROUND COUNTER). Argument of routine W.AMMO(WEAPON AMMUNITION) carrying the value of the TANK updating.

PROGRAM LISTING

```

1  ROUTINE BLU.CREATE
2  ''
3  DEFINE I AS AN INTEGER VARIABLE
4  PRINT 1 LINE THUS
   ROUTINE BLU.CREATE
5  ''
6  READ WMIN,WMAX,WSTREAM
7  SKIP 2 CARDS
8  READ PMIN,PMAX,PSTREAM
9  SKIP 2 CARDS
10 READ CMIN,CMAX,CSTREAM

```



```

11 SKIP 2 CARDS
12 LIST WMIN,WMAX,WSTREAM,PMIN,PMAX,PSTREAM
13 LIST CMIN,CMAX,CSTREAM
14 ''
15 READ CNUM,PNUM,SNUM
16 SKIP 2 CARDS
17 LIST PNUM,CNUM,SNUM
18 CALL PARAMETERS
19 LET N.COMPANY.COMMANDER = CNUM
20 CREATE EVERY COMPANY.COMMANDER
21 FOR EVERY COMPANY.COMMANDER, DO
22   READ N.CCL.V.ITEMS (COMPANY.COMMANDER)
23   READ S4.OFF (COMPANY.COMMANDER)
24   LIST N.CCL.V.ITEMS (COMPANY.COMMANDER)
25   LIST S4.OFF (COMPANY.COMMANDER)
26 LOOP
27 SKIP 2 CARDS
28 LET N.PLATOON.LEADER = PNUM
29 CREATE EVERY PLATOON.LEADER
30 FOR EVERY PLATOON.LEADER, DO
31   READ PCO.CDR (PLATOON.LEADER)
32   READ N.PCL.V.ITEMS (PLATOON.LEADER)
33   LIST PCO.CDR (PLATOON.LEADER)
34   LIST N.PCL.V.ITEMS (PLATOON.LEADER)
35   FILE PLATOON.LEADER
36   IN CO.UNIT (PCO.CDR (PLATOON.LEADER))
37 LOOP
38 SKIP 2 CARDS
39 LET N.SUPPLY.OFFICER = SNUM
40 CREATE EVERY SUPPLY.OFFICER
41 FOR EVERY SUPPLY.OFFICER, DO
42   READ SCO.CDR (SUPPLY.OFFICER)
43   READ N.SCL.V.ITEMS (SUPPLY.OFFICER)
44   LIST SCO.CDR (SUPPLY.OFFICER)
45   LIST N.SCL.V.ITEMS (SUPPLY.OFFICER)
46   SKIP 3 CARDS
47 LOOP
48 CALL BASIC.LOAD
49 SKIP 2 CARDS
50 ''
51 READ N.TANKS
52                                     SKIP 3 CARDS
53 FOR I = 1 TO N.TANKS, DO
54   CREATE A TANK
55   READ NAME (TANK), COLOR (TANK), SYS.TYPE (TANK),
56   WPN.TYPE (TANK), AP.TOW (TANK), HE.DRAG (TANK),
57   AW1.OR.MSL3 (TANK), AW2.OR.ADM (TANK),
58   AMMO5 (TANK), AMMO6 (TANK),
59   SLOAD1 (TANK), SLOAD2 (TANK), SLOAD3 (TANK),
60   SLOAD4 (TANK), SLOAD5 (TANK), SLOAD6 (TANK),
61   OH1 (TANK), OH2 (TANK), OH3 (TANK), OH4 (TANK), OH5 (TANK),
62   OH6 (TANK), TAC1 (TANK), TAC2 (TANK), TAC3 (TANK),
63   TAC4 (TANK), TAC5 (TANK), TAC6 (TANK),
64   PLTLDR (TANK), COCDR (TANK), SUPOFF (TANK),
65   MAX.WT (TANK), MAX.CUBE (TANK)
66   IF SYS.TYPE (TANK) EQ 7
67     FILE TANK IN S.UNIT (SUPOFF (TANK))
68   ALWAYS
69   IF SYS.TYPE (TANK) NE 7
70     FILE TANK IN PLAT.UNIT (PLTLDR (TANK))
71     SCHEDULE AN UP.W.AMMO (TANK) IN 1 MINUTE
72   ALWAYS
73   FILE TANK IN TNK.ALIVE
74   LET POINTER (TANK) = TANK
75   LET TCU (TANK) = MAX.WT (TANK)
76   LET TWT (TANK) = MAX.CUBE (TANK)
77   SKIP 2 CARDS
78 LOOP
79 RETURN

```


80 END

EXPLANATION OF CODE

Line 3 Defines recursive variables for the routine.

Line 4 Prints message to mark the start of the routine.

Lines 6-13 Read and print out data establishing min and max times entities will check their ammo status as well as random number streams for calculations.

Lines 15-17 Establish the number of company commanders, platoon leaders, and supply officers to be created in the simulation and print out the information input.

Line 18 Calls routine PARAMETERS.

Lines 19-27 Create the specified number of company commanders, and read the attributes of each and print the information.

Lines 29-38 Create the specified number of platoon leaders, read their attributes, file them in appropriate sets and print the information.

Lines 40-48 Create the specified number of supply officers, read their attributes, file them in appropriate sets and print the information.

Line 49 Calls routine BASIC.LOAD.

Line 52 Reads the number of TANKS to be created for the simulation.

Lines 53-78 Create the specified number of TANKs read their attributes, file them in appropriate sets and print the information.

Line 71 Schedules an UP.W.AMMO event for each TANK to initially set ammunition status.

D. "ROUTINE PARAMETERS"

Routine PARAMETERS is called from routine BLU.CREATE to reserve space for and read values into the following arrays: WPNLON, COMLON, NOMEN, ROF, POF, and POD. Additionally the critical cutoff values for supply action (C.L2WT, C.L2C, C.L2PCT, and C.L1PCT), the trip times to and from the supply points (MINTRIP and MAXTRIP), and the random number streams TSTREAM and RSTREAM are established.

GLOBAL_VARIABLES_(ALPHA)

NOMEN(NOMENCLATURE). Array specifying the name of rounds played.

GLOBAL_VARIABLES_(INTEGER)

CNUM(COMPANY NUMBER). Specifies the number of COMPANY.COMMANDERS created.

RCODE(RESUPPLY CODE) (2-d). Holds the pointer value for a supply unit's SCL.V.ITEMs(SUPPLY CLASS V ITEMS).

ROF(RATE OF FIRE) (2-d). Specifies the rates of fire for the six weapons of any weapon system.

RSTREAM(RESUPPLY STREAM). Specifies the random number stream to be used for resupply calculations.

TSTREAM(TRIP RANDOM NUMBER STREAM).

GLOBAL_VARIABLES_(REAL)

C.L1PCT(CRITICAL LON1 PERCENTAGE). Holds the maximum percentage that LON1 requisitions may be reduced to in order to release space on a convoy for other critical LON1 and LON2 requests.

C.L2CU(CRITICAL LON2 CUBE). Holds the maximum percent of cube LON2 requisitions may be cut to in order to release space on a convoy for other critical LON1 and LON2 requisitions.

C.L2PCT(CRITICAL LON2 PERCENTAGE). Holds the maximum percentage that LON2 requisitions may be reduced to in order to release space on a convoy for other critical LON1 and LON2 requests.

C.L2.WT(CRITICAL LON2 WEIGHT). Holds the maximum percent of cube LON2 requisitions may be cut to in order to release

space on a convoy for other critical LON1 and LON2 requisitions.

COMLON(2-d) (COMPANY LEVEL OF NEED).

MAXTRIP(MAX TRIP). The maximum time required for a convoy to reach its intended destination.

MINTRIP(MIN TRIP). The minimum time required for a convoy to reach its intended destination.

PLTLON(2-d) (PLATOON LEVEL OF NEED).

POD(2-d) (PROBABILITY OF DAMAGE). For all weapon systems played.

POF(2-d) (PROBABILITY OF FIRE). For all weapon systems played.

WPNLON(2-d) (WEAPON LEVEL OF NEED).

RECURSIVE VARIABLES (INTEGER)

MAX.CL.V.ITEMS(MAX NUMBER OF CLASS V ITEMS PLAYED).

N.WPN.TYPES(NUMBER OF WEAPON TYPES).

PROGRAM LISTING

```
1 ROUTINE PARAMETERS
2 ''
3 PRINT 1 LINE THUS
  ROUTINE PARAMETERS
4 DEFINE MAX.CL.V.ITEMS, AND N.WPN.TYPES
5 AS AN INTEGER VARIABLES
6 READ MAX.CL.V.ITEMS
7 LIST MAX.CL.V.ITEMS
8 SKIP 3 CARDS
9 ''
10 RESERVE WPNLON(*,*) AS MAX.CL.V.ITEMS BY 5
11 READ WPNLON
12 SKIP 3 CARDS
13 LIST WPNLON
14 RESERVE PLTLON(*,*) AS MAX.CL.V.ITEMS BY 5
15 READ PLTLON
16 SKIP 3 CARDS
```



```

17 LIST PLTLON
18 RESERVE COMLON (*,*) AS MAX.CL.V.ITEMS BY 5
19 READ COMLON
20 SKIP 2 CARDS
21 LIST COMLON
22 ''
23 RESERVE NOMEN (*) AS MAX.CL.V.ITEMS
24 READ NOMEN
25 SKIP 2 CARDS
26 LIST NOMEN
27 ''
28 READ N.WPN.TYPES LIST N.WPN.TYPES
29 ''
30 SKIP 3 CARDS
31 RESERVE ROF AS N.WPN.TYPES BY 6
32 READ ROF
33 SKIP 3 CARDS
34 LIST ROF
35 ''
36 RESERVE POF AS N.WPN.TYPES BY 6
37 READ POF
38 SKIP 3 CARDS
39 LIST POF
40 ''
41 RESERVE POD AS N.WPN.TYPES BY 4
42 READ POD
43 SKIP 2 CARDS
44 LIST POD
45 ''
46 '' RESERVE AN ARRAY TO HOLD RESUPPLY REQUESTS
47 RESERVE RCODE (*,*) AS CNUM BY 6
48 ''
49 '' SET UP CRITICAL CUTOFF VALUES FOR S-4
50 READ C.L2WT,C.L2CU,C.L2PCT,C.L1PCT
51 LIST C.L2WT,C.L2CU,C.L2PCT,C.L1PCT
52 SKIP 2 CARDS
53 READ MINTRIP,MAXTRIP,TSTREAM,RSTREAM
54 LIST MINTRIP,MAXTRIP,TSTREAM,RSTREAM
55 SKIP 2 CARDS
56 ''
57 RETURN
58 END

```

EXPLANATION OF CODE

Line 3 Prints a message specifying the start of the routine.

Lines 4-5 Define recursive variables for the routine.

Lines 6-7 Read and print out the max number of ammunition types to be played in the simulation.

Lines 10-21 Reserve space for and read the threshold values for the level of need played at the weapon, platoon, and company levels in the simulation.

Lines 23-26 Reserve an array for and read in the
nomenclatures played.

Line 28 Reads the number of weapon types played.

Lines 31-44 Reserve space for, assign values to, and print
out the arrays used in the battle computations. These
arrays are: Rate of Fire; Probability of Fire; and
Probability of Damage.

Lines 46-47 Reserve space to hold values of resupply
requisitions created by each company in the simulation.

Lines 49-51 Read in the critical cutoff values for resupply
action. These values are: Critical LON2 weight;
Critical LON2 cube; Critical LON2 Percent; and Critical
LON1 percent.

Lines 53-54 Read in the min and max times required to
travel between the supply point and the company.

E. "ROUTINE BASIC.LOAD"

Routine BASIC.LOAD is called from routine BLU.CREATE to
create the base ammunition assets of all platoons,
companies, and supply officers in the model.

EVENTS_CALLED

UP.COM.AMMO(UPDATE COMPANY AMMO).

UP.PLT.AMMO(UPDATE PLATOON AMMO).

GLOBAL_VARIABLES_(ALPHA)

NOMEN(NOMENCLATURE). Specifies name of round.

GLOBAL_VARIABLES_(INTEGER)

CCODE(COMPANY CODE) (2-d). Holds the pointer value for a

company's CCL.V.ITEMS(COMPANY CLASS V ITEMS).

CNUM(COMPANY NUMBER). Specifies the number of

COMPANY.COMMANDERS created.

PCODE(PLATOON CODE) (2-d). Holds the pointer value for a

platoon's PCL.V.ITEMS(PLATOON CLASS V ITEMS).

PNUM(PLATOON NUMBER).

SCODE(SUPPLY CODE) (2-d). Holds the pointer value for a

supply unit's SCL.V.ITEMS(SUPPLY CLASS V ITEMS).

SNUM(NUMBER OF SUPPLY OFFICERS).

PERMANENT_ATTRIBUTES_(INTEGER)

N.CCL.V.ITEMS. Number of company class V items(UNIQUE TO A
COMPANY.COMMANDER).

N.PCL.V.ITEMS. Number of platoon class V items(UNIQUE TO A
PLATOON.LEADER).

N.SCL.V.ITEMS. Number of supply class V items(UNIQUE TO A
SUPPLY.OFFICER).

RECURSIVE_VARIABLES_(INTEGER)

C,I,P,S - Loop indicies.

SETS_USED

CO.AMMO (COMPANY AMMUNITION) . Owned by a COMPANY.COMMANDER.

Members are the unit's CCL.V.ITEMS (COMPANY CLASS V
ITEMS)

PLT.AMMO (PLATOON AMMUNITION) . Owned by a PLATOON.LEADER.

Members are the unit's PCL.V.ITEMS (PLATOON CLASS V
ITEMS)

S.AMMO (SUPPLY AMMUNITION) . Owned by a SUPPLY.OFFICER.

Members are the unit's SCL.V.ITEMS (SUPPLY CLASS V ITEMS)

TEMPORARY ENTITIES

CCL.V.ITEM. (COMPANY CLASS V ITEM) .

PCL.V.ITEM. (PLATOON CLASS V ITEM) .

SCL.V.ITEM. (SUPPLY CLASS V ITEM) .

TEMPORARY ATTRIBUTES (ALPHA)

C.RND.CNTR (COMPANY ROUND COUNTER) . Argument for event
UP.COM.AMMO (UPDATE COMPANY AMMUNITION) holding a
pointer of a company unit.

CNOMEN (COMPANY NOMENCLATURE) . This attribute of a
CCL.V.ITEM (COMPANY CLASS V ITEM) contains the name of
the particular ammunition.

PNOMEN (PLATOON NOMENCLATURE) . This attribute of a
PCL.V.ITEM (PLATOON CLASS V ITEM) contains the name of
the particular ammunition.

SNOMEN (SUPPLY NOMENCLATURE). This attribute of a

SCL.V.ITEM (SUPPLY CLASS V ITEM) contains the name of the particular ammunition.

TEMPORARY_ATTRIBUTES_(INTEGER)

CAC (COMPANY AMMUNITION CODE) .

ONHAND (INTEGER). This attribute of a SCL.V.ITEM (SUPPLY CLASS V ITEM) holds the on-hand balance of stocks for an ammunition.

P.RND.CNTR (PLATOON ROUND COUNTER). Argument of the event UP.PLT.AMMO (UPDATE PLATOON AMMUNITION) carrying the pointer value of the platoon currently updating.

PAC (PLATOON AMMUNITION CODE) . Of a PCL.V.ITEM (PLATOON CLASS V ITEM)

RDS.PKG (ROUNDS PER PACKAGE). Number packed per pallet of an SCL.V.ITEM (SUPPLY CLASS V ITEM) .

SAC (SUPPLY AMMUNITION CODE) .

TEMPORARY_ATTRIBUTES_(REAL)

CU.PKG (CUBE PACKAGE) . Cube of ammo pallet for an SCL.V.ITEM (SUPPLY CLASS V ITEM) .

WT.PKG (WEIGHT PACKAGE) . Weight of ammo pallet for an SCL.V.ITEM (SUPPLY CLASS V ITEM) .

PROGRAM_LISTING

```
1 ROUTINE BASIC.LOAD
2 DEFINE I,P,S,AND C AS INTEGER VARIABLES
```



```

3 PRINT 1 LINE THUS
  ROUTINE BASIC.LOAD
4 ''
5 ''SETUP PLATOON BASIC LOADS
6 RESERVE PCODE(*,*) AS PNUM BY *
7 FOR P = 1 TO PNUM, DO
8   RESERVE PCODE(PNUM,*) AS N.PCL.V.ITEMS (P)
9   FOR I = 1 TO N.PCL.V.ITEMS (P), DO
10    CREATE A PCL.V.ITEM CALLED PCODE (P,I)
11    LET PAC(PCODE(P,I)) = I
12    LET PNOMEN(PCODE(P,I)) = NOMEN(I)
13    FILE PCODE(P,I) IN 'PLT.AMMO (P)
14  LOOP
15  SCHEDULE A UP.PLT.AMMO (P) IN 1 MINUTE
16 LOOP
17 ''
18 ''SETUP COMPANY BASIC LOADS
19 RESERVE CCODE(*,*) AS CNUM BY *
20 FOR C=1 TO CNUM, DO
21   LIST C,CNUM,N.CCL.V.ITEMS (C)
22   RESERVE CCODE(CNUM,*) AS N.CCL.V.ITEMS (C)
23   FOR I = 1 TO N.CCL.V.ITEMS (C), DO
24    CREATE A CCL.V.ITEM CALLED CCODE (C,I)
25    LET CAC(CCODE(C,I)) = I
26    LET CNOMEN(CCODE(C,I)) = NOMEN(I)
27    FILE CCODE(C,I) IN 'CO.AMMO (C)
28  LOOP
29  SCHEDULE AN UP.COM.AMMO(C) IN 1 MINUTE
30 LOOP
31 ''
32 '' SETUP SUPPLY OFFICER BASIC LOADS
33 RESERVE SCODE(*,*) AS SNUM BY *
34 FOR S=1 TO SNUM, DO
35   LIST S,SNUM,N.SCL.V.ITEMS (S)
36   RESERVE SCODE(SNUM,*) AS N.SCL.V.ITEMS (S)
37   FOR I = 1 TO N.SCL.V.ITEMS (S), DO
38    CREATE A SCL.V.ITEM CALLED SCODE (S,I)
39    LET SAC(SCODE(S,I)) = I
40    LET SNOMEN(SCODE(S,I)) = NOMEN(I)
41    READ WT.PKG(SCODE(S,I)), CU.PKG(SCODE(S,I))
42    READ RDS.PKG(SCODE(S,I)), ONHAND(SCODE(S,I))
43    FILE SCODE(S,I) IN 'S.AMMO (S)
44  LOOP
45 LOOP
46 RETURN
47 END

```

EXPLANATION OF CODE

Line 2 Defines recursive variables used in the routine.

Line 3 Prints a message marking the beginning of the routine.

Line 6 Reserves the first index of the ragged array PCODE as the number of platoons played in the simulation.

PCODE will eventually hold the ammunition types used by the platoon.

Line 7 Begins the outside loop over each Platoon.Leader for the code segment which creates and stores the ammunition carried by each platoon. Loop ends on line 16.

Line 8 Reserves the second index of the ragged array PCODE to hold the number of ammo types carried by each platoon.

Line 9 Begins the inside loop over each ammunition type carried by a platoon. Loop ends on line 14.

LINE 10-13 Create each ammo type carried by a platoon, record its ammunition code and nomenclature, and file the ammo type in the platoon leader's ammo stocks (PLT.AMMO).

Line 15 Schedules the initial update for the platoon.

Line 19 Reserves the first index of the ragged array CCODE to the number of companies played in the simulation. CCODE will eventually hold the ammunition types used by the company.

Line 20 Begins the outside loop over each company commander, the code segment which creates and stores the ammunition carried by each company. Loop ends on line 30.

Line 22 Reserves the second index of the ragged array CCODE to hold the number of ammo types carried by each company.

Line 23-28 Begin the inside loop over each ammunition type carried by a company. Loop ends on line 28.

Line 24-27 Create each ammo type carried by a company; record its ammunition code and nomenclature; and file the ammo type in the company ammo stocks (CO.AMMO).

Line 33 Reserves the first index of the ragged array SCODE to the number of supply elements played in the simulation.

Line 34 Begins the outside loop over each supply officer, the code segment which creates and stores the ammunition carried by each supply unit. Loop ends on line 45.

Line 36 Reserves the second index of the ragged array SCODE to hold the number of ammo types carried by each supply unit.

Line 37 Begins the inside loop over each ammunition type carried by a supply unit. Loop ends on line 44.

Lines 38-43 Create each ammo type carried by a supply unit; record its ammunition code, nomenclature, standard package weight, standard package cube, number of rounds

per package, starting level, and file them in the supply unit's ammo stocks (S.AMMO) .

F. "ROUTINE W.AMMO"

Routine W.AMMO is called by events UP.W.AMMO, CO.RESUPPLY.ARR, and REDISTRIBUTE for each TANK in the model in order to update each TANK's ammunition LON. An argument, NO.BATTLE, is set to indicate whether battle information should be obtained or if only a simple update is required. Additionally a program indicator WFLAG, is set to provide information on the system's status.

ARGUMENTS USED

A - Identifies the weapon system updating its ammunition.

NO.BATTLE. Indicates whether routine BATTLE should be called.

"0" indicates no

"1" indicates yes

WFLAG. Weapon status indicator. If WFLAG=100 the weapon is out of a particular ammunition this signals the platoon to do an immediate update. If WFLAG=1 the TANK has been knocked out of battle and should no longer update its ammunition status.

DEFINE TO MEAN

AMMO1. AP.TOW(ARMOR PIERCING/TOW ROUNDS).

AMMO2. HE.DRAG (HEAT/Dragon rounds) .

AMMO3. AW1.OR.MSL3 (ALTERNATE WEAPON 1 OR MISSILE 3) .

AMMO4. AW2.OR.ADM (ALTERNATE WEAPON 2 OR AIR DEFENSE
MISSILE) .

GLOBAL VARIABLES

WPNLON (2-d) (WEAPON LEVEL OF NEED) .

PERMANENT ATTRIBUTE (REAL)

TIME.V

RECURSIVE VARIABLES (INTEGERS)

I - Loop index.

TEMP.LON (TEMPORARY LON). Place holder for LON computations.

TRY Routing indicator to the ammo type being updated.

RECURSIVE VARIABLES (REAL)

PCT (PERCENT) .

ROUTINES CALLED

BATTLE

TEMPORARY ATTRIBUTES (INTEGER)

AMMO5 (AMMUNITION 5) . Of a TANK.

AMMO6 (AMMUNITION 6) . Of a TANK.

AP.TOW (ARMOR-PIERCING/TOW) . Ammunition 1.

TAC1. TANK AMMUNITION CODE 1.

TAC2. TANK AMMUNITION CODE 2.

TAC3. TANK AMMUNITION CODE 3.

TAC4. TANK AMMUNITION CODE 4.

TAC5. TANK AMMUNITION CODE 5.

TAC6. TANK AMMUNITION CODE 6.

AW1.OR.MSL3(ALTERNATE WEAPON 1 OR MISSILE 3). Ammunition 3
of a TANK.

AW2.OR.ADM(ALTERNATE WEAPON 2 OR AIR DEFENSE MISSILE).

Ammunition 4 of a TANK.

FKILL(FIREPOWER KILL). Indicates whether a TANK has
sustained a firepower kill during the battle.

"0" indicates no

"1" indicates yes

HE.DRAG(HEAT/DRAGON ROUNDS). Ammunition 2 of a TANK.

KKILL(CATASTROPHIC KILL). Indicates whether a TANK has
sustained a catastrophic kill during the battle.

"0" indicates no

"1" indicates yes

MKILL(MOBILITY KILL). Indicates whether a TANK has sustained
a mobility kill during the battle.

"0" indicates no

"1" indicates yes

MPKILL(MOBILITY AND FIREPOWER KILL). Indicates whether a
TANK has sustained a mobility and firepower kill during
the battle.

"0" indicates no

"1" indicates yes

OH1(ON-HAND 1). Current balance of ammunition 1 on a TANK.

OH2 (ON-HAND 2) . Current balance of ammunition 2 on a TANK.

OH3 (ON-HAND 3) . Current balance of ammunition 3 on a TANK.

OH4 (ON-HAND 4) . Current balance of ammunition 4 on a TANK.

OH5 (ON-HAND 5) . Current balance of ammunition 5 on a TANK.

OH6 (ON-HAND 6) . Current balance of ammunition 6 on a TANK.

POINTER Machine address of a TANK.

PLTLDR (PLATOON LEADER) Of TANK.

SLOAD1 (STOWED LOAD 1) . Optimal load ammo type 1.

SLOAD2 (STOWED LOAD 2) . Optimal load ammo type 2.

SLOAD3 (STOWED LOAD 3) . Optimal load ammo type 3.

SLOAD4 (STOWED LOAD 4) . Optimal load ammo type 4.

SLOAD5 (STOWED LOAD 5) . Optimal load ammo type 5.

SLOAD6 (STOWED LOAD 6) . Optimal load ammo type 6.

WLON1 (WEAPON LEVEL OF NEED 1) . Urgency of need ammunition 1.

WLON2 (WEAPON LEVEL OF NEED 2) . Urgency of need ammunition 2.

WLON3 (WEAPON LEVEL OF NEED 3) . Urgency of need ammunition 3.

WLON4 (WEAPON LEVEL OF NEED 4) . Urgency of need ammunition 4.

WLON5 (WEAPON LEVEL OF NEED 5) . Urgency of need ammunition 5.

WLON6 (WEAPON LEVEL OF NEED 6) . Urgency of need ammunition 6.

TEMPORARY ENTITIES

TANK

PROGRAM LISTING

1 ROUTINE W.AMMO GIVEN A,AND NO.BATTLE YIELDING WFLAG


```

2  DEFINE NO.BATTLE,WFLAG,TEMP.LON AS INTEGER VARIABLES
3  DEFINE PCT AS A REAL VARIABLE
4  DEFINE A,TRY, AND I AS AN INTEGER VARIABLES
5  PRINT 2 LINES WITH TIME.V AND A AS FOLLOWS
    EVENT W.AMMO CALLED AT TIME.V **.*
    WEAPON SYSTEM ***** UPDATING
6  ''
7  IF NO.BATTLE EQ 0,
8      CALL BATTLE(A)
9  ALWAYS
10 IF FKILL(A) EQ 1 OR KILL(A) EQ 1 OR MKILL(A) EQ 1
11 OR MPKILL(A) EQ 1
12     PRINT 2 LINES WITH POINTER(TANK) THUS
        BATTLE DAMAGE ON TANK ***** PREVENTS
        AMMO ASSESSMENT. SEQUENCE ENDED.
13     LET WFLAG = 1 '' BATTLE DAMAGE SUSTAINED
14     RETURN
15 OTHERWISE
16     ''
17     ''CAPTURE CURRENT INFORMATION FOR PLT.COMPUTATIONS.
18     LET OH1(A) = AMMO1(A)
19     LET OH2(A) = AMMO2(A)
20     LET OH3(A) = AMMO3(A)
21     LET OH4(A) = AMMO4(A)
22     LET OH5(A) = AMMO5(A)
23     LET OH6(A) = AMMO6(A)
24     ''
25     ''BEGIN LON COMPUTATIONS
26     LET PCT= AMMO1(A)/SLOAD1(A)
27     LET I= TAC1(A)
28     LET TRY = 1
29     GO TO CHECK
30     '1'LET WLON1(A)=TEMP.LON
31     ''
32     LET PCT= AMMO2(A)/SLOAD2(A)
33     LET I= TAC2(A)
34     LET TRY = 2
35     GO TO CHECK
36     '2'LET WLON2(A)=TEMP.LON
37     ''
38     LET PCT= AMMO3(A)/SLOAD3(A)
39     LET I= TAC3(A)
40     LET TRY = 3
41     GO TO CHECK
42     '3'LET WLON3(A)=TEMP.LON
43     ''
44     LET PCT= AMMO4(A)/SLOAD4(A)
45     LET I= TAC4(A)
46     LET TRY = 4
47     GO TO CHECK
48     '4'LET WLON4(A)=TEMP.LON
49     ''
50     LET PCT= AMMO5(A)/SLOAD5(A)
51     LET I= TAC5(A)
52     LET TRY = 5
53     GO TO CHECK
54     '5'LET WLON5(A)=TEMP.LON
55     ''
56     LET PCT= AMMO6(A)/SLOAD6(A)
57     LET I= TAC6(A)
58     LET TRY = 6
59     GO TO CHECK
60     '6'LET WLON6(A)=TEMP.LON
61     GO TO 7
62     ''
63     'CHECK'
64     IF PCT GE WPNLON(I,1),
65         LET TEMP.LON= 5

```



```

66      GO TO ROUTING
67      ALWAYS
68      ''
69      IF PCT GE WPNLON(I,2) ,
70          LET TEMP.LON= 4
71          GO TO ROUTING
72      ALWAYS
73      ''
74      IF PCT GE WPNLON(I,3) ,
75          LET TEMP.LON= 3
76          GO TO ROUTING
77      ALWAYS
78      ''
79      IF PCT GE WPNLON(I,4) ,
80          LET TEMP.LON= 2
81          GO TO ROUTING
82      ALWAYS
83      ''
84      IF PCT GE WPNLON(I,5) ,
85          LET TEMP.LON= 1
86          LET WFLAG = 100
87      ALWAYS
88      'ROUTING' GO TO 1,2,3,4,5,6 PER TRY
89      '7'
90      RETURN
91      END

```

EXPLANATION OF CODE

Lines 2-4 Define recursive variables used in the routine.

Line 5 Prints a message marking the start of the event and identifying the weapon system updating.

Lines 7-9 Check if battle information should be obtained, and call routine BATTLE to obtain this current information.

Lines 10-15 Check if battle damage has been sustained, print a message if damage has been incurred and end the routine without action.

Lines 17-23 Update the current knowledge of the ammo situation on the weapon system.

Lines 25-61 Sequentially address each ammo type carried on a weapon system; determine a percent fill; temporarily transfer control to 'CHECK' (line 63) in order to determine the correct LON to be assigned; and assign an LON.

Lines 63-89 Receive a percent value from lines 25-61 and match this percent to that type of weapon's LON array. Pass the LON value determined back to lines 25-61 for assignment.

G. "ROUTINE P.CLASS.V"

Routine P.CLASS.V is called by events UP.PLT.AMMO, CO.RESUPPLY.ARR, and REDISTRIBUTE for each platoon played in the model to update their ammunition LONs according to the latest weapon LON information. Additionally, a program indicator, PFLAG, is set to provide information on the platoon's status.

ARGUMENTS

J - Argument for routine P.CLASS.V specifying the platoon currently updating.

PFLAG. Platoon status indicator:

PFLAG = 1 - indicates that the platoon is out of stock for an ammo type and signals the company to update its ammo status.

PFLAG = N.PCL.VITEMS - indicates that the platoon has no need for more ammo. That is, the platoon's weapons have been destroyed or damaged.

GLOBAL_VARIABLES_(INTEGER)

PCODE(PLATOON CODE) (2-d). Holds the pointer value for a platoon's PCL.V.ITEMS(PLATOON CLASS V ITEMS).

TANK

PLTLON(PLATOON LEVEL OF NEED) (2-d).

PERMANENT_ATTRIBUTES_(REAL)

TIME.V

PERMANENT_ATTRIBUTES_(INTEGER)

N.PCL.V.ITEMS.(NUMBER OF PLATOON CLASS V ITEMS). Unique to a PLATOON.LEADER.

RECURSIVE_VARIABLES_(INTEGER)

I - Loop index.

RECURSIVE_VARIABLES_(REAL)

PCT(PERCENT). Place holder for calculations.

ROUTINE_CALLED

P.CLASS.V

SETS_USED

PLAT.UNIT(PLATOON UNIT). Owned by a PLATOON.LEADER. Members are the unit's combat vehicles(TANKs).

TEMPORARY_ATTRIBUTES_(INTEGER)

TAC1. TANK AMMUNITION CODE 1.

TAC2. TANK AMMUNITION CODE 2.

TAC3. TANK AMMUNITION CODE 3.

TAC4. TANK AMMUNITION CODE 4.

TAC5. TANK AMMUNITION CODE 5.

TAC6. TANK AMMUNITION CODE 6.

FKILL(FIREPOWER KILL). Indicates whether a TANK has
sustained a firepower kill during the battle.

"0" indicates no

"1" indicates yes

KKILL(CATASTROPHIC KILL). Indicates whether a TANK has
sustained a catastrophic kill during the battle.

"0" indicates no

"1" indicates yes

MKILL(MOBILITY & FIREPOWER KILL). Indicates whether a TANK
has sustained a mobility & firepower kill during the
battle.

"0" indicates no

"1" indicates yes

MKILL(MOBILITY KILL). Indicates whether a TANK has sustained
a mobility kill during the battle.

"0" indicates no

"1" indicates yes

OH1(ON-HAND 1). Current balance of ammunition 1 on a TANK.

OH2(ON-HAND 2). Current balance of ammunition 2 on a TANK.

OH3(ON-HAND 3). Current balance of ammunition 3 on a TANK.

OH4 (ON-HAND 4) . Current balance of ammunition 4 on a TANK.

OH5 (ON-HAND 5) . Current balance of ammunition 5 on a TANK.

OH6 (ON-HAND 6) . Current balance of ammunition 6 on a TANK.

P.SHORT (PLATOON SHORTAGE) . Current shortage of a PCL.V.ITEM
(PLATOON CLASS V ITEM) . Unique to each platoon and ammo
type.

P.CMBT.LOSS (PLATOON COMBAT LOSS) . Indicates that the loss of
a platoon weapon system negates the need for this ammo
type.

PAMMO.LON (PLATCON AMMO LEVEL OF NEED) . Unique for each
platoon and ammo type.

PCURR.LOAD (PLATOON CURRENT LOAD) . On-hand balance of ammo.
Unique for each platoon and weapon type.

PL.B.LOAD (PLATOON BASIC LOAD) . Optimal load for a PCL.V.ITEM
(PLATOON CLASS V ITEM) .

SLOAD1 (STOWED LOAD 1) . Optimal load ammo type 1.

SLOAD2 (STOWED LOAD 2) . Optimal load ammo type 2.

SLOAD3 (STOWED LOAD 3) . Optimal load ammo type 3.

SLOAD4 (STOWED LOAD 4) . Optimal load ammo type 4.

SLOAD5 (STOWED LOAD 5) . Optimal load ammo type 5.

SLOAD6 (STOWED LOAD 6) . optimal load ammo type 6.

PROGRAM LISTING

1 ROUTINE P.CLASS.V GIVEN J YIELDING PFLAG


```

2  DEFINE I,PFLAG,AND J AS INTEGER VARIABLES
3  DEFINE PCT AS A REAL VARIABLE
4  PRINT 1 LINE WITH TIME.V AS FOLLOWS
   EVENT PLT.AMMO CALLED AT TIME.V **.*
5  FOR T=1 TO N.PCL.V.ITEMS(J), DO
6      LET PCURR.LOAD(PCODE(J,I)) = 0
7      LET PL.B.LOAD(PCODE(J,I)) = 0
8      FOR EVERY TANK IN PLAT.UNIT(J)
9          WITH MKILL(TANK) EQ 0 AND KILL(TANK) EQ 0
10             AND MFKILL(TANK) EQ 0, DO
11  ..
12      IF TAC1(TANK)=I
13          ADD OH1(TANK) TO PCURR.LOAD(PCODE(J,I))
14          IF FKILL(TANK) EQ 0, 'NO FKILL
15              ADD SLOAD1(TANK) TO PL.B.LOAD(PCODE(J,I))
16          ALWAYS
17          GO TO SHORTAGE
18      OTHERWISE
19  ..
20      IF TAC2(TANK)=I
21          ADD OH2(TANK) TO PCURR.LOAD(PCODE(J,I))
22          IF FKILL(TANK) EQ 0,
23              ADD SLOAD2(TANK) TO PL.B.LOAD(PCODE(J,I))
24          ALWAYS
25          GO TO SHORTAGE
26      OTHERWISE
27  ..
28      IF TAC3(TANK)=I
29          ADD OH3(TANK) TO PCURR.LOAD(PCODE(J,I))
30          IF FKILL(TANK) EQ 0,
31              ADD SLOAD3(TANK) TO PL.B.LOAD(PCODE(J,I))
32          ALWAYS
33          GO TO SHORTAGE
34      OTHERWISE
35  ..
36      IF TAC4(TANK)=I
37          ADD OH4(TANK) TO PCURR.LOAD(PCODE(J,I))
38          IF FKILL(TANK) EQ 0,
39              ADD SLOAD4(TANK) TO PL.B.LOAD(PCODE(J,I))
40          ALWAYS
41          GO TO SHORTAGE
42      OTHERWISE
43  ..
44      IF TAC5(TANK)=I
45          ADD OH5(TANK) TO PCURR.LOAD(PCODE(J,I))
46          IF FKILL(TANK) EQ 0,
47              ADD SLOAD5(TANK) TO PL.B.LOAD(PCODE(J,I))
48          ALWAYS
49          GO TO SHORTAGE
50      OTHERWISE
51  ..
52      IF TAC6(TANK)=I
53          ADD OH6(TANK) TO PCURR.LOAD(PCODE(J,I))
54          IF FKILL(TANK) EQ 0,
55              ADD SLOAD6(TANK) TO PL.B.LOAD(PCODE(J,I))
56          ALWAYS
57      ALWAYS
58  ..
59  'SHORTAGE'
60      LET P.SHORT(PCODE(J,I)) = PL.B.LOAD(PCODE(J,I))
61                               - PCURR.LOAD(PCODE(J,I))
62  LOOP
63  ..
64  'CHECK FOR BATTLEDAMAGE
65  IF PL.B.LOAD(PCODE(J,I)) EQ 0,
66      LET PFLAG = PFLAG+1
67      LET P.SHORT(PCODE(J,I)) = 0
68  IF P.CMBT.LOSS(PCODE(J,I)) EQ 0,

```



```

69      LET P.CMBT.LOSS(PCODE(J,I)) = 1
70      PRINT 2 LINES WITH I, J THUS
        BATTLE DAMAGE HAS NEGATED THE NEED FOR
        AMMO ** IN PLATOON **
71      ALWAYS
72      CYCLE
73      OTHERWISE
74      ''
75      LET PCT= PCURR.LOAD(PCODE(J,I))
76      /PL.B.LOAD(PCODE(J,I))
77      ''
78      IF PCT GE PLTLON(I,1),
79      LET PAMMO.LON(PCODE(J,I))=5
80      CYCLE
81      ALWAYS
82      IF PCT GE PLTLON(I,2),
83      LET PAMMO.LON(PCODE(J,I))=4
84      CYCLE
85      ALWAYS
86      IF PCT GE PLTLON(I,3),
87      LET PAMMO.LON(PCODE(J,I))=3
88      CYCLE
89      ALWAYS
90      IF PCT GE PLTLON(I,4),
91      LET PAMMO.LON(PCODE(J,I))=2
92      CYCLE
93      ALWAYS
94      IF PCT GE PLTLON(I,5),
95      LET PAMMO.LON(PCODE(J,I))=1
96      LET PFLAG = 100
97      ALWAYS
98      LOOP
99      RETURN
100     END

```

EXPLANATION OF CODE

Lines 2-3 Define recursive variables used in the routine.

Line 4 Prints out a message to mark the start of the routine.

Line 5 Starts the outside loop over all the ammo types carried by the platoon. Loop ends on line 98.

Lines 6-7 Zero out the platoon on-hand ammunition and basic load ammunition for the update.

Lines 8-10 Begin the inner loop over all TANKs in the platoon in order to obtain the most current information

available on each TANK. The loop ends on line 62.

Note: the information obtained from each TANK is

"current knowledge" from its on-hand attributes. This

is what the TANK "knows" it has on-hand. Opposite to

this is "perfect knowledge" from its AMMO1-AMMO6

attributes, which is what the TANK actually has on-hand.

Lines 12-57 Cycle the index over the six possible choices

on the TANK until identification or lack of

identification of the ammo code being considered is

made. If identification is made, the on-hand balance of

the TANK's known load is added to the platoon balance.

If the weapon system is not F-KILLED the base load of

the TANK's ammo is added to the platoon base load. Base

load is used in determining how much should be ordered.

If the weapon has been F-KILLED the weapon's base load

is ignored. This lowers the amount of ammo the platoon

needs to have on-hand to keep its systems full. After

updating, control is transferred to the shortage loop,

lines 59-60

Lines 59-61 Provide a running update of the amount of ammo

the platoon is short as the update continues over all

the weapon systems in the platoon.

Line 62 Closes the loop for a TANK, transferring control either to line 8 to evaluate the next TANK for this ammo type or out to complete the LON computation for the ammo type.

Lines 64-73 Check for a zero base load balance in the platoon's ammo. A new zero balance indicates that battle damage to platoon weapons has negated the need for further requisitioning of that type of ammo. A message is printed identifying the round type. Further requisitions for this ammo type would not be made.

Lines 75-76 Compute the percentage of fill (on-hand/base load) for an ammo type.

Lines 78-97 Cycle the percentage over the five possible LON threshold values until the correct value is found and assigned.

Line 98 Closes the major loop in the routine and transfers control back to line 5 in order to update the next ammo type.

H. "ROUTINE COM.AMMO"

Routine COM.AMMO is called by events UP.COM.AMMO and CO.RESUPPLY.ARR for each company played in the model to update ammunition LONs according to the most recent platoon

LON information. An argument, NO.BATTLE, is set to indicate whether RES.REQS should be filed or whether a simple update is sought. Additionally, a program indicator, CFLAG, is set to provide information on the system's status.

ARGUMENTS_USED

C - Argument of COM.AMMO (COMPANY AMMUNITION) holding the pointer value of the company currently updating.

CFLAG. Company status indicator; CFLAG = N.CL.V.ITEM

indicates that the company has no need for ammunition.

That is, the company's weapons have been damaged or destroyed.

NO.BATTLE. Indicates whether RES.REQS should be filed.

"0" indicates no

"1" indicates yes

EVENTS_SCHEDULED

UP.S4.AMMO (UPDATE S-4 AMMUNITION).

GLOBAL_VARIABLES_(ALPHA)

NOMEN (NOMENCLATURE). Specifies name of round.

GLOBAL_VARIABLES_(INTEGERS)

CCODE (COMPANY CODE) (2-d). Holds the pointer value for a company's CCL.V.ITEMs (COMPANY CLASS V ITEMS).

PCODE (PLATOON CODE) (2-d). Holds the pointer value for a platoon's PCL.V.ITEMs (PLATOON CLASS V ITEMS).

PLATOON.LEADER

RCODE(RESUPPLY CODE) (2-d) . Holds the pointer value for a

resupply unit's SCL.V.ITEMS(SUPPLY CLASS V ITEMS) .

TSTREAM(TRIP RANDOM NUMBER STREAM)

COMLON (2-d) (COMPANY LEVEL OF NEED) .

GLOBAL VARIABLES (REAL)

MAXTRIP(MAX TRIP) . The maximum time required for a convoy to reach its intended destination.

MINTRIP(MIN TRIP) . The minimum time required for a convoy to reach its intended destination.

PERMANENT ATTRIBUTES (REAL)

TIME.V

PERMANENT ATTRIBUTES (INTEGER)

N.CCL.V.ITEMS(NUMBER OF COMPANY CLASS V ITEMS) . Unique to a Unit.

REQN(REQUISITION) .

S4.OFF(Company S-4 OFFICER) .

RECURSIVE VARIABLES (INTEGER)

COUNT. Indicates if a resupply requisition has been filed.

I - Loop index.

RECURSIVE VARIABLES (REAL)

PCT(PERCENT) .

ROUTINES CALLED

UNIFORM.F

SETS_USED

CO.UNIT (COMPANY UNIT) . Owned by a COMPANY.COMMANDER.

Members are the unit's PLATOON.LEADERS.

CWANT.LIST (COMPANY WANT LIST) . Contains a listing of the resupply requests outstanding for the company.

TEMPORARY_ENTITIES

RES.REQ (RESUPPLY REQUEST) .

TEMPORARY_ATTRIBUTES_(ALPHA)

RNOMEN (RESUPPLY NOMENCLATURE) .

TEMPORARY_ATTRIBUTES_(INTEGER)

C.CMBT.LOSS (COMPANY COMBAT LOSS) . Indicates that the loss of a company weapon system negates the need for this ammo type.

C.NUM.REQN (NUMBER OF COMPANY REQUISITIONS) . Total number of RES.REQs submitted for an SCL.V.ITEM.

C.SHORT (COMPANY SHORTAGE) . Total rounds short for a type ammo.

CAMMO.LON (COMPANY AMMUNITION LEVEL OF NEED) .

CCURR.LOAD (COMPANY CURRENT LOAD) . On-hand balance for an ammo type.

CO.B.LOAD (COMPANY BASIC LOAD) . Optimal load for an ammo type.

ISSUEE. Argument of UP.S4.AMMO (UPDATE.S4.AMMUNITION) holding the pointer of the requesting unit.

ISSUER. Argument of UP.S4.AMMO (UPDATE.S4.AMMUNITION) holding the pointer of the unit supply officer.

PCURR.LOAD (PLATOON CURRENT LOAD). On-hand balance for an ammo type.

PL.B.LOAD (PLATOON BASIC LOAD). Optimal load for an ammo type.

RAC (RESUPPLY AMMO CODE). Of a RES.REQ (RESUPPLY REQUEST).

REQUESTOR. Of a RES.REQ (RESUPPLY REQUEST). Contains the pointer of the unit requesting.

RQTY (REQUIRED QUANTITY). Of a RES.REQ (RESUPPLY REQUEST).

SPRIORITY (SUPPLY PRIORITY). Of a RES.REQ (RESUPPLY REQUEST).

TEMPORARY ATTRIBUTES (REAL)

TIME. Of creation of request.

PROGRAM LISTING

```
1  ROUTINE COM.AMMO GIVEN C,NO.BATTLE  YIELDING CFLAG
2  DEFINE NO.BATTLE,I,COUNT,CFLAG,AND C
3  AS INTEGER VARIABLES
4  DEFINE PCT AS A REAL VARIABLE
5  PRINT 1 LINE WITH TIME.V AS FOLLOWS
6  EVENT COM.AMMO CALLED AT TIME.V **.***
7  ,,
8  FOR I=1 TO N.CCL.V.ITEMS(C),DO
9    LET CCURR.LOAD(CCODE(C,I)) = 0
10   LET CO.B.LOAD(CCODE(C,I)) = 0
11   FOR EVERY PLATOON.LEADER IN CO.UNIT(C),DO
12     ADD PCURR.LOAD(PCODE(PLATOON.LEADER,I))
13     TO CCURR.LOAD(CCODE(C,I))
14     ADD PL.B.LOAD(PCODE(PLATOON.LEADER,I))
15     TO CO.B.LOAD(CCODE(C,I))
16   LOOP
17   IF CO.B.LOAD(CCODE(C,I)) = 0,
18     LET CFLAG = CFLAG + 1
19     IF C.CMBT.LOSS(CCODE(C,I)) = 0
20       LET C.CMBT.LOSS(CCODE(C,I)) = 1
```



```

20      PRINT 2 LINES WITH I, C THUS
        BATTLE DAMAGE HAS NEGATED THE NEED FOR
        AMMO *** IN COMPANY ***
21      ALWAYS
22      CYCLE
23      OTHERWISE
24      LET PCT=CCURR.LOAD(CCODE(C,I))
25      /CO.B.LOAD(CCODE(C,I))
26      IF PCT GE COMLON(I,1)
27      LET CAMMO.LON(CCODE(C,I))=5
28      GO TO SHORTAGE
29      ALWAYS
30      IF PCT GE COMLON(I,2)
31      LET CAMMO.LON(CCODE(C,I))=4
32      GO TO SHORTAGE
33      ALWAYS
34      IF PCT GE COMLON(I,3)
35      LET CAMMO.LON(CCODE(C,I))=3
36      GO TO SHORTAGE
37      ALWAYS
38      IF PCT GE COMLON(I,4)
39      LET CAMMO.LON(CCODE(C,I))=2
40      GO TO SHORTAGE
42      ALWAYS
43      IF PCT GE COMLON(I,5)
44      LET CAMMO.LON(CCODE(C,I))=1
45      ALWAYS
46      'SHORTAGE'
47      LET C.SHORT(CCODE(C,I)) = CO.B.LOAD(CCODE(C,I))
48      - CCURR.LOAD(CCODE(C,I))
49
50      'RESUPPLY'
51      'CHECK IF RESUPPLY NEEDED
52      IF CAMMO.LON(CCODE(C,I)) GE 5 OR NO.BATTLE EQ 1
53      GO TO LOOP1
54      OTHERWISE
55
56      'IF A PRIOR LON IS EQUAL TO THE CURR LON, CYCLE.
57
58      FOR EACH RES.REQ IN CWANT.LIST(C)
59      WITH RAC(RES.REQ)=I AND REQUESTOR(RES.REQ)=C, DO
60      IF SPRIORITY(RES.REQ) = CAMMO.LON(CCODE(C,I))
61      GO TO LOOP1
62      OTHERWISE
63      'LOOP3' LOOP
64
65      'ORDER THE AMMUNITION
66      CREATE A RES.REQ CALLED RCODE(C,I)
67      LET COUNT = 1
68      LET REQUESTOR(RCODE(C,I)) = C
69      LET RPNT(RCODE(C,I)) = RCODE(C,I)
70      LET ROTY(RCODE(C,I)) = CO.B.LOAD(CCODE(C,I))
71      - CCURR.LOAD(CCODE(C,I))
72      LET STATUS(RCODE(C,I)) = "TOS4"
73      LET RAC(RCODE(C,I)) = I
74      LET RNOMEN(RCODE(C,I)) = NOMEN(I)
75      LET SPRIORITY(RCODE(C,I)) = CAMMO.LON(CCODE(C,I))
76      LET TIME(RCODE(C,I)) = TIME.V
77      LET REQN(C) = REQN(C) + 1
78      LET C.NUM.REQN(CCODE(C,I)) =
79      C.NUM.REQN(CCODE(C,I)) + 1
80      FILE RCODE(C,I) IN CWANT.LIST(C)
81
82      'LOOP1' LOOP
83
84      IF COUNT = 1,
85      SCHEDULE A'UP.S4.AMMO(S4.OFF(C),C)
86

```



```
87     IN UNIFORM.F (MINTRIP,MAXTRIP,TSTREAM) MINUTES
88     ALWAYS
89     RETURN
90     END
```

EXPLANATION OF CODE

Lines 2-4 Define recursive variables used in the routine.

Line 5 Prints out a message to mark the start of the routine.

Line 7 Starts the outside loop over all the ammo types carried by the company. Loop ends on line 83.

Lines 8-9 Zero out the company on-hand ammunition and basic load ammunition for the update.

Lines 10-15 Begin an inner loop over all platoons in the company in order to obtain the most current on-hand and base load information available in each platoon. The loop ends on line 15. Note: the information obtained from each platoon is "current knowledge", which is what the platoon knows it has on-hand, as opposed to "perfect knowledge", which is what the platoon actually has on-hand.

Lines 16-23 Check for a zero base load balance in the company's ammo. This would indicate that battle damage to company weapons has negated the need for further requisitioning of that type of ammo. A message is

printed identifying the round type. Further requisitions would not be made.

Line 24-25 Compute the percentage of fill (on-hand/base load) for an ammo type.

Lines 26-45 Cycle the percentage over the five possible LON threshold values until the correct value is found and assigned. Control is then directed to lines 47-49 to determine the unit shortage.

Lines 47-49 Provide an update of the total amount of ammo the company is short.

Lines 51-55 Check if an initial ammo request needs to be filed. Ammunitions with LON values equal to 5 are cycled. Additionally, a NO.BATTLE check is made. NO.BATTLE equal to 1 indicates that a simple update is required and no RES.REQS are to be submitted.

Lines 57-64 Check any previous requisitions for the ammo type being reviewed. If the previous request has a priority equal to the current LON there is no need for a new request to be filed and the routine is cycled.

Lines 66-80 Create a requisition to be forwarded to battalion supply. The attributes set in the request are: requestor identification; a request pointer; a

requested quantity; a request status; a request ammo code pointing to a particular ammo type; a request nomenclature; a request priority; time submitted; and total requests submitted for that ammo type.

Line 81 Files the request in the company's want list.

Line 83 Closes the major loop in the routine and transfers control to Line 7 to begin a loop over the next ammo type.

Lines 85-88 The variable COUNT keys the fact that a requisition has been created and must be forwarded to battalion. If appropriate, an UP.S4.AMMO is scheduled.

I. "ROUTINE BATTLE"

Routine BATTLE is called from routine W.AMMO to obtain the most recent expenditure and damage information. The routine generates ammunition expenditures for each alive weapon according to designated rates of fire and probabilities of fire. It also randomly damages and destroys vehicles according to user designated probabilities of damage.

ARGUMENTS USED

A - Holds the value of the weapon system updating its situation.

DEFINE TO MEAN

AMMO1. AP.TOW (ARMOR PIERCING/TOW ROUNDS) .
AMMO2. HE.DRAG (HEAT/DRAGON ROUNDS) .
AMMO3. AW1.OR.MSL3 (ALTERNATE WEAPON 1 OR MISSILE 3) .
AMMO4. AW2.OR.ADM (ALTERNATE WEAPON 2 OR AIR DEFENSE
MISSILE) .

GLOBAL_VARIABLES_(INTEGER)

ROF (RATE OF FIRE) (2-d) . Specifies the rates of fire for the
six weapons on board any weapon system.
RSTREAM (RESUPPLY STREAM) .
WSTREAM (WEAPON RANDOM NUMBER STREAM) .

GLOBAL_VARIABLES_(REAL)

B.END (BATTLE END) . Termination time of daily battle.
B.START (BATTLE START) . Start time of daily battle.
POD (2-d) (PROBABILITY OF DAMAGE) . For all weapon systems.
POF (2-d) (PROBABILITY OF FIRE) . For all weapon systems.

PERMANENT_ATTRIBUTE_(REAL)

TIME.V

RECURSIVE_VARIABLES_(REAL)

LAM1 (LAMBDA 1) . Holds the value of the probability of fire
for AMMO1 from the POF (PROBABILITY OF FIRE) array.
LAM2 (LAMBDA 2) . Holds the value of the probability of fire
for AMMO2 from the POF (PROBABILITY OF FIRE) array.

LAM3(LAMBDA 3) . Holds the value of the probability of fire
for AMMO3 from the POF(PROBABILITY OF FIRE) array.

LAM4(LAMBDA 4) . Holds the value of the probability of fire
for AMMO4 from the POF(PROBABILITY OF FIRE) array.

LAM5(LAMBDA 5) . Holds the value of the probability of fire
for AMMO5 from the POF(PROBABILITY OF FIRE) array.

LAM6(LAMBDA 6) . Holds the value of the probability of fire
for AMMO6 from the POF(PROBABILITY OF FIRE) array.

ROUTINES CALLED

EXPONENTIAL.F, MAX.F, and RANDOM.F .

SETS

TNK.ALIVE(TANK ALIVE) . Owned by the system. Members are
vehicles still alive within the simulation.

TEMPORARY_ATTRIBUTES_(INTEGER)

AMMO5(AMMUNITION 5) . Of TANK.

AMMO6(AMMUNITION 6) . Of TANK.

AP.TOW(ARMOR-PIERCING/TOW) . Ammunition 1 of TANK.

AW1.OR.MSL3(ALTERNATE WEAPON 1 OR MISSILE 3) . Ammunition 3
of TANK.

AW2.OR.ADM(ALTERNATE WEAPON 2 OR AIR DEFENSE MISSILE) .
Ammunition 4 of TANK.

FKILL(FIREPOWER KILL). Indicates whether a TANK has sustained a firepower kill during the battle.

"0" indicates no

"1" indicates yes

HE.DRAG(HEAT/DRAGON ROUNDS). Ammunition 2 of TANK.

KKILL(CATASTROPHIC KILL). Indicates whether a TANK has sustained a catastrophic kill during the battle.

"0" indicates no

"1" indicates yes

MFKILL(MOBILITY & FIREPOWER KILL). Indicates whether a TANK has sustained a mobility and firepower kill during the battle.

"0" indicates no

"1" indicates yes

MKILL(MOBILITY KILL). Indicates whether a TANK has sustained a mobility kill during the battle.

"0" indicates no

"1" indicates yes

POINTER. Machine address of TANK.

WPN.TYPE(WEAPON TYPE). Of TANK.

PROGRAM LISTING

```
1 ROUTINE BATTLE(A)
2 DEFINE A AS AN INTEGER VARIABLE
3 DEFINE LAM1,LAM2,LAM3,LAM4,
4 LAM5,AND LAM6 AS REAL VARIABLES
5 ''
6 ''CHECK IF BATTLE IN PROGRESS, IF NOT, RETURN
7 PRINT 1 LINE WITH E.START AND B.END AS FOLLOWS
   BSTART = **.**** B.END = **.****
8 IF TIME.V LT B.START OR TIME.V GT B.END, RETURN
9 OTHERWISE
10 ''
11 '' ESTABLISH RATE OF FIRE FOR EACH AMMUNITION FIRED.
12 LET LAM1 = ROF(WPN.TYPE(A),1)
```



```

13 LET LAM2 = ROF(WPN.TYPE(A),2)
14 LET LAM3 = ROF(WPN.TYPE(A),3)
15 LET LAM4 = ROF(WPN.TYPE(A),4)
16 LET LAM5 = ROF(WPN.TYPE(A),5)
17 LET LAM6 = ROF(WPN.TYPE(A),6)
18 ''
19 '' ESTABLISH AMMUNITION EXPENDITURES FOR EACH AMMO
20 IF RANDOM.F(RSTREAM) LE POF(WPN.TYPE(A),1)
21 LET AMMO1(A) = MAX.F(AMMO1(A)
22 - EXPONENTIAL.F(LAM1*TIME.V,WSTREAM),0)
23 ALWAYS
24 IF RANDOM.F(RSTREAM) LE POF(WPN.TYPE(A),2)
25 LET AMMO2(A) = MAX.F(AMMO2(A)
26 - EXPONENTIAL.F(LAM2*TIME.V,WSTREAM),0)
27 ALWAYS
28 IF RANDOM.F(RSTREAM) LE POF(WPN.TYPE(A),3)
29 LET AMMO3(A) = MAX.F(AMMO3(A)
30 - EXPONENTIAL.F(LAM3*TIME.V,WSTREAM),0)
31 ALWAYS
32 IF RANDOM.F(RSTREAM) LE POF(WPN.TYPE(A),4)
33 LET AMMO4(A) = MAX.F(AMMO4(A)
34 - EXPONENTIAL.F(LAM4*TIME.V,WSTREAM),0)
35 ALWAYS
36 IF RANDOM.F(RSTREAM) LE POF(WPN.TYPE(A),5)
37 LET AMMO5(A) = MAX.F(AMMO5(A)
38 - EXPONENTIAL.F(LAM5*TIME.V,WSTREAM),0)
39 ALWAYS
40 IF RANDOM.F(RSTREAM) LE POF(WPN.TYPE(A),6)
41 LET AMMO6(A) = MAX.F(AMMO6(A)
42 - EXPONENTIAL.F(LAM6*TIME.V,WSTREAM),0)
43 ALWAYS
44 ''
45 '' DETERMINE IF DAMAGES HAVE BEEN SUSTAINED
46 IF RANDOM.F(RSTREAM) LE POD(WPN.TYPE(A),1)
47 LET MKILL(A) = 1
48 PRINT 1 LINES THUS
49 BATTLE DAMAGE SUSTAINED
50 LIST POINTER(A),MKILL(A)
51 ALWAYS
52 IF RANDOM.F(RSTREAM) LE POD(WPN.TYPE(A),2)
53 LET FKILL(A) = 1
54 PRINT 1 LINES THUS
55 BATTLE DAMAGE SUSTAINED
56 LIST POINTER(A),FKILL(A)
57 ALWAYS
58 IF RANDOM.F(RSTREAM) LE POD(WPN.TYPE(A),3)
59 LET MFKILL(A) = 1
60 PRINT 1 LINE THUS
61 BATTLE DAMAGE SUSTAINED
62 LIST POINTER(A),MFKILL(A)
63 REMOVE A FROM TNK.ALIVE
64 ALWAYS
65 IF RANDOM.F(RSTREAM) LE POD(WPN.TYPE(A),4)
66 LET KILL(A) = 1
67 PRINT 1 LINE THUS
68 BATTLE DAMAGE SUSTAINED
69 LIST POINTER(A),KILL(A)
70 REMOVE A FROM TNK.ALIVE
71 ALWAYS
72 RETURN
73 END

```

EXPLANATION OF CODE

Lines 2-4 Define recursive variables used in the routine.

Lines 6-9 Check if a battle is currently in progress, if not it causes the routine to return without action.

Lines 11-17 Assign a rate of fire by weapon and ammunition type for the six ammo types carried by the weapon updating. Rates of fire are obtained from a rate of fire array input by the user.

Lines 19-43 Establish ammo expenditures for each of the six ammo types carried by a weapon system. Use of the weapon during the time period is established by comparing a random number to a probability of fire for the weapon system. Expenditure of ammo is then computed through the use of an exponential function using the weapon's rate of fire as λ .

Lines 45-71 Determine if battle damage has been sustained during the time period. Evaluation is made by comparing a random number against the various types of kill possible.

J. "ROUTINE UP.DATE"

Routine UP.DATE is called from event B.UP.DATE and is used to print a battle summary listing weapon and unit assets, as well as supply status.

GLOBAL_VARIABLES_(INTEGER)

CCL.V.ITEM (COMPANY CLASS V ITEM) .

COMPANY.COMMANDER

CONVOY

CNUM (COMPANY NUMBER) . Specifies the number of

COMPANY.COMMANDERS created.

PCL.V.ITEM (PLATOON CLASS V ITEM) .

PLATOON.LEADER

PNUM (PLATOON NUMBER) .

RES.REQ. (RESUPPLY REQUEST) .

SCL.V.ITEM. (SUPPLY CLASS V ITEM) .

SNUM (NUMBER OF SUPPLY OFFICERS) .

SUPPLY.OFFICER

T.CGO (TRUCK CARGO) .

TANK

PERMANENT ATTRIBUTES

TIME.V

RECURSIVE VARIABLES

I, J, K - Loop indices.

SETS

CARGO

CO.AMMO (COMPANY AMMUNITION) . Owned by a COMPANY.COMMANDER.

Members are the unit's CCL.V.ITEMS (COMPANY CLASS V
ITEMS) .

CWANT.LIST (COMPANY WANT LIST) .

PLAT.UNIT (PLATOON UNIT) . Owned by a PLATOON.LEADER. Members
are the unit's combat vehicles (TANKS) .

PLT.AMMO (PLATOON AMMUNITION) . Owned by a PLATOON.LEADER.
Members are the unit's PCL.V.ITEMS (PLATOON CLASS V
ITEMS) .

S.AMMO (SUPPLY AMMUNITION) . Owned by a SUPPLY.OFFICER.
Members are the unit's SCL.V.ITEMS (SUPPLY CLASS V
ITEMS) .

S.UNIT (SUPPLY UNIT) . Owned by a SUPPLY.OFFICER. Members are
the unit's supply vehicles (TANKS) .

SCONVOY (SUPPLY CONVOY) .

SREQN.LIST (SUPPLY REQUISITION LIST) .

SWANT.LIST (SUPPLY WANT LIST) . Owned by a SUPPLY.OFFICER.

Members are RES.REQS (RESUPPLY REQUESTS) from the combat
units.

TNK.ALIVE (TANK ALIVE) .

TEMPORARY_ATTRIBUTES_(INTEGER)

SYS.TYPE (SYSTEM TYPE) .

PROGRAM_LISTING

```
1 ROUTINE UP.DATE
2 DEFINE I,J,AND K AS INTEGER VARIABLES
3 PRINT 3 LINES WITH TIME.V AS FOLLOWS
  ''
  ROUTINE UP.DATE CALLED AT **.*
  ''
4 ''
5 LIST ATTRIBUTES OF EACH COMPANY.COMMANDER
```



```

6  FOR J = 1 TO CNUM, DO
7    LIST ATTRIBUTES OF EACH CCL.V.ITEM IN CO.AMMO(J)
8    LIST ATTRIBUTES OF EACH RES.REQ IN CWANT.LIST(J)
9  LOOP
10 ''
11 LIST ATTRIBUTES OF EVERY PLATOON.LEADER
12 FOR I = 1 TO PNUM, DO
13 LIST ATTRIBUTES OF EACH PCL.V.ITEM IN PLT.AMMO(I)
14 LIST ATTRIBUTES OF EACH TANK IN PLAT.UNIT(I)
15 LOOP
16 ''
17 LIST ATTRIBUTES OF EVERY SUPPLY.OFFICER
18 FOR K = 1 TO SNUM, DO
19   LIST ATTRIBUTES OF EACH SCL.V.ITEM IN S.AMMO(K)
20   LIST ATTRIBUTES OF EACH RES.REQ IN SWANT.LIST(K)
21   LIST ATTRIBUTES OF EACH RES.REQ IN SREQN.LIST(K)
22   LIST ATTRIBUTES OF EACH CONVOY IN SCONVOY(K)
23   LIST ATTRIBUTES OF EACH TANK IN S.UNIT(K)
24 LOOP
25 ''
26 FOR EACH TANK IN TNK.ALIVE
27 WITH SYS.TYPE(TANK) EQ 7, DO
28   LIST ATTRIBUTES OF EACH T.CGO IN CARGO(TANK)
29 LOOP
30 ''
31 PRINT 2 LINES WITH TIME.V AS FOLLOWS
    ''
    ROUTINE UP.DATE ENDED AT **.****
32 RETURN
33 END

```

EXPLANATION OF CODE

- Line 2 Defines recursive variables for the routine.
- Line 3 Prints a message marking the start of the routine
- Lines 5-9 Mark a loop over the attributes and sets
belonging to each company commander.
- Line 5 Lists attributes of each company commander.
- Line 7 Lists attributes of each class V item the company
commander owns.
- Line 8 Lists attributes of each resupply request the
company commander has outstanding.
- Lines 11-15 Mark a loop over the attributes and sets
belonging to each platoon leader.
- Line 11 Lists attributes of each PLATOON.LEADER.

Line 13 Lists the attributes of each class V item the platoon leader owns.

Line 14 Lists the attributes of each weapon system of the unit.

Lines 17-24 Mark a loop over the attributes and sets belonging to each supply officer.

Line 17 Lists the attributes of each supply officer

Line 19 Lists the attributes of each class V item

Line 20 Lists the attributes of outstanding requests from supported companies.

Line 21 Lists the attributes of outstanding requests sent to the ATP/ASP.

Line 22 Lists the attributes of each convoy the supply unit owns.

Line 23 Lists the attributes of all supply vehicles in the unit.

Lines 26-29 Mark a loop over all resupply vehicles, listing the attributes of all supply cargo carried.

K. "ROUTINE FILE.UP.DATE"

This routine is called from event UP.S4.AMMO. It is used to update the S-4's outstanding request list by filing

new requests by priority and eliminating duplicate older requests of lesser priority.

ARGUMENT

A - Points to supply officer updating.

COM. Points to company currently updating.

GLOBAL_VARIABLE_(INTEGER)

SCODE(SUPPLY CODE) (2-d). HOLD'S POINTER FOR SCL.V.ITEMS.

PERMANENT_ATTRIBUTE_(INTEGER)

N.SWANT.LIST(NUMBER IN SUPPLY WANT) .

RECURSIVE_VARIABLE_(INTEGER)

NEW.REQ(NEW REQUISITION) .

OLD.REQ(OLD REQUISITION) .

ROUTINES_CALLED

FILE.UPDATE

SETS_USED

CWANT.LIST(COMPANY WANT LIST) .

SWANT.LIST(SUPPLY WANT LIST) .

TEMPORARY_ATTRIBUTES_(INTEGER)

DEMAND. for SCL.V.ITEM.

M.C.CGO.LIST(MEMBER CONVOY CARGO LIST) . System generated

variable indicating whether or not a resupply request is
on a convoy cargo list.

RAC (RESUPPLY AMMUNITION CODE). Attribute of an SCL.V.ITEM
which points to a specific ammo type carried by the
supply unit.

REQUESTOR. Attribute of a RES.REQ pointing to the requesting
unit.

RQTY (RESUPPLY QUANTITY). Attribute of a RES.REQ holding the
total quantity requested by a unit.

SPRIORITY (SUPPLY PRIORITY). Attribute of a RES.REQ
specifying the urgency of need for the ammo request.

TEMPORARY_ATTRIBUTE (ALPHA)

RNOMEN (RESUPPLY NOMENCLATURE). Attribute of a RES.REQ
specifying the requested ammo's nomenclature.

TEMPORARY_ENTITY

RES.REQ (RESUPPLY REQUEST).

SCL.V.ITEM (SUPPLY CLASS V ITEM). Holds information for a
supply unit about a particular ammo type used by the
unit. Belongs to the set S.AMMO.

PERMANENT_ATTRIBUTE (DOUBLE)

TIME.V. Simulation time.

PROGRAM LISTING

```
1 ROUTINE FILE.UPDATE(A,COM)
2 DEFINE A,COM,OLD.REQ,NEW.REQ AS INTEGER VARIABLES
3 '' FILE THE REQUEST CURRENTLY BEING RECEIVED
4 '' AND CAPTURE DEMAND DATA
5 PRINT 1 LINE WITH TIME.V THUS
  ROUTINE FILE.UPDATE CALLED AT TIME.V = **.***
6 FOR ALL NEW.REQ IN CWANT.LIST(COM)
7 WITH M.C.CGO.LIST(NEW.REQ)=0, DO
```



```

8      IF NEW.REQ IS NOT IN SOME SWANT.LIST
9          LET DEMAND(SCODE(A,RAC(NEW.REQ))) =
10         DEMAND(SCODE(A,RAC(NEW.REQ)))+ RQTY(NEW.REQ)
11     ALWAYS
12     FOR ALL OLD.REQ IN SWANT.LIST(A)
13         WITH RNOMEN(OLD.REQ) = RNOMEN(NEW.REQ)
14         AND M.C.CGO.LIST(OLD.REQ) = 0
15         AND REQUESTOR(OLD.REQ) = REQUESTOR(NEW.REQ), DO
16             IF SPRIORITY(OLD.REQ) NE SPRIORITY(NEW.REQ),
17                 LET DEMAND(SCODE(A,RAC(NEW.REQ))) =
18                 DEMAND(SCODE(A,RAC(OLD.REQ)))
19                 + RQTY(NEW.REQ) - RQTY(OLD.REQ)
20                 REMOVE OLD.REQ FROM SWANT.LIST(A)
21                 REMOVE OLD.REQ FROM CWANT.LIST(COM)
22                 DESTROY RES.REQ CALLED OLD.REQ
23     ALWAYS
24     LOOP
25     IF NEW.REQ IS NOT IN SOME SWANT.LIST,
26         FILE NEW.REQ IN SWANT.LIST(A)
27     ALWAYS
28     LOOP
29     RETURN
31     END

```

EXPLANATION OF CODE

Line 2 Defines the recursive variables used in the routine.

Line 5 Prints a message marking the beginning of the event.

Lines 6-7 Begin the major outer loop for the routine by

looping over all outstanding requests that have not been loaded on a resupply truck. Loop ends on line 28.

Lines 8-11 Determine if the request has been filed

previously with the S-4. If not it is treated as new and its full demand data is captured.

Lines 12-25 Begin an inner loop which determines if a

similar request for the type of ammo in the outer loop

has been filed previously with the S-4. If so, a

comparison is made between request priority values. If

the values are the same, this indicates that the new

request has been filed previously and no further action is necessary. If the priorities are different, this indicates that the unit is increasing its urgency of need for the ammo. The new request is filed and the old one is destroyed. The difference between the new and old values is added to the demand. Loop ends on line 24.

Line 24 Closes the inner loop for the routine. Control is transferred back to line 12 to continue comparing requests or out to the next new request.

Lines 25-27 File all genuinely new requests in the S-4's Want list.

Line 28 Closes the main routine loop begun on line 8. Control is transferred back to the next request or out.

L. "ROUTINE LOAD.THE.TRUCKS"

This routine is called from event UP.S4.AMMO. It is used to load supply cargo onto individual trucks. In execution, the event checks the number of trucks allocated to carry a specific supply request and loads the number of trucks specified subject to the weight and cube limitations of the truck.

ARGUMENT (INTEGER)

A - Pointer value to the S-4 officer.

CON.ID (CONVOY ID) . Holds the pointer value of the convoy currently being filled.

N.RNDS (NUMBER OF ROUNDS) .

RR (RESUPPLY REQUEST) . Holds the pointer of the resupply request being currently filled.

ARGUMENT (REAL)

CU.REQ (CUBE REQUIRED) . Holds the total cube that is required in order to ship a resupply request.

WT.REQ (WEIGHT REQUIRED) . Holds the total weight that is required in order to ship a resupply request.

GLOBAL VARIABLE (INTEGER)

RES.REQ (RESUPPLY REQUEST) .

SCODE (SUPPLY CODE) .

TANK

RECURSIVE VARIABLES (INTEGER)

RC.TEMP (ROUND/CUBE TEMPORARY) . Holds the computational value of the number of rounds that may be loaded on a truck due to cube restrictions.

RNDS (ROUNDS) . Holds the number of rounds being released to fill a request.

RW.TEMP (ROUND/WEIGHT TEMPORARY) . Holds the computational value of the number of rounds that may be loaded on a truck due to weight restrictions.

T.COUNT(TRUCK COUNT). Holds the value of the number of trucks already loaded for a particular resupply request.

RECURSIVE_VARIABLE__(REAL)

PCT(PERCENT).

ROUTINES

LOAD.THE.TRUCK, MIN.F, TRUNC.F

SETS_USED

CARGO. Contains a listing of the T.CGO loaded on a particular truck.

ELEMENTS. Set of trucks owned by a CONVOY.

S.UNIT(SUPPLY UNIT). Contains the unit's supply vehicles(TANKS).

TEMPORARY_ATTRIBUTES_(ALPHA)

RNOMEN(RESUPPLY NOMENCLATURE). Attribute of an SCL.V.ITEM containing the name of the particular ammo type.

TNOMEN(TRUCK NOMENCLATURE). Contains the name of a particular ammo type.

TEMPORARY_ATTRIBUTES_(INTEGER)

FKILL(FIREPOWER KILL).

KKILL(CATASTROPHIC KILL).

M.ELEMENTS. Specifies membership in a CONVOY.

MFKILL(MOBILITY AND FIREPOWER KILL).

MKILL(MOBILITY KILL).

N.CARGO(NUMBER IN CARGO). Indicates the total number of
T.CGO items loaded on a particular truck.

N.ELEMENTS(NUMBER OF ELEMENTS). In a convoy.

N.T.ALLOC(NUMBER OF TRUCKS ALLOCATED). Contains the number
of trucks to be allocated to move the rounds released
for a resupply request.

ONHAND. Holds the on-hand balance for rounds of a particular
type at the resupply point.

POINTER. Of a TANK.

RAC(RESUPPLY AMMUNITION CODE). Attribute of an SCL.V.ITEM
which points to a specific ammo type carried by the
supply unit.

RDS.PKG(ROUNDS PACKAGE). Attribute of a SCL.V.ITEM
specifying the number of rounds on an ammo pallet.

RFILL(RESUPPLY FILL). Attribute of a RES.REQ specifying the
amount of ammunition released to fill a request.

RQTY(RESUPPLY QUANTITY). Attribute of a RES.REQ holding the
total quantity requested by a unit.

RRPNTR(RESUPPLY REQUEST POINTER).

SPACE. Attribute of a CONVOY holding the amount of loading
space available on trucks in the convoy.

TCU(TRUCK CUBE) . The cube that a truck has available to be filled.

TPNTR(TRUCK POINTER) .

TQTY(TRUCK QUANTITY) . Holds the number of rounds within a T.CGO that are loaded on a truck.

TRAC(TRUCK AMMUNITION CODE) . Of a T.CGO item.

TWT(TRUCK WEIGHT) . The weight that a truck has available to be filled.

TEMPORARY_ATTRIBUTES_(REAL)

CU.PKG(CUBE PACKAGE) Attribute of a SCL.V.ITEM specifying the cube of an ammunition pallet.

WT.PKG(WEIGHT PACKAGE) . Attribute of a SCL.V.ITEM specifying the weight of a pallet of the ammo being considered.

TEMPORARY_ENTITY

T.CGO. Supplies carried on a resupply vehicle (TANK) .

PROGRAM_LISTING

```
1 ROUTINE LOAD THE TRUCKS
2 (A,RR,WT.REQ,CU.REQ,N.RNDS,CON.ID)
3 DEFINE A,RR,N.RNDS,CON.ID,T.COUNT,RW.TEMP,
4 RC.TEMP,RNDS AS INTEGER VARIABLES
5 DEFINE WT.REQ,CU.REQ,PCT AS REAL VARIABLES
6 PRINT 1 LINE THUS
7 ROUTINE LOAD THE TRUCKS CALLED
8 LET T.COUNT = 0
9 FOR EVERY TANK IN S.UNIT(A)
10 WITH M.ELEMENTS(TANK) EQ 0
11 AND MKILL(TANK) EQ 0 AND FKILL(TANK) EQ 0
12 AND MFKILL(TANK) EQ 0 AND KKILL(TANK) EQ 0, DO
13 'LOOP CHECK
14 LET T.COUNT = T.COUNT +1
15 IF T.COUNT GT N.T.ALLOC(RES.REQ) OR WT.REQ LE 0
16 OR CU.REQ LE 0 OR N.RNDS LE 0
17 LEAVE
18 OTHERWISE
19 'CHECK IF ITEMS EXCEED THE TRUCK'S CAPACITY
20 'IF SO, ADJUST
```



```

20 IF WT.REQ GT TWT(TANK) OR CU.REQ GT TCU(TANK)
21 LET RW.TEMP = TWT(TANK)
22 /WT.PKG(SCODE(A,RAC(RES.REQ)))
23 * RDS.PKG(SCODE(A,RAC(RES.REQ)))
24 LET RC.TEMP = TCU(TANK)
25 /CU.PKG(SCODE(A,RAC(RES.REQ)))
26 * RDS.PKG(SCODE(A,RAC(RES.REQ)))
27 LET RNDS = MIN.F(RW.TEMP,RC.TEMP)
28 LET N.RNDS = N.RNDS - RNDS
29 ELSE
30 LET RNDS = N.RNDS
31 LET SPACE(CON.ID) = 1
32 ALWAYS
33 CREATE A T.CGO
34 LET TPNTR(T.CGO) = POINTER(TANK)
35 LET RRPNTR(T.CGO) = RES.REQ
36 LET TNOMEN(T.CGO) = RNOMEN(RES.REQ)
37 LET TRAC(T.CGO) = RAC(RES.REQ)
38 LET TOTY(T.CGO) = RNDS
39 FILE T.CGO IN CARGO(TANK)
40 ''REDUCE THE OUTSTANDING QUANTITY ON THE RES.REQ
41 LET N.RNDS = N.RNDS - RNDS
42 LET RFILL(RES.REQ) = RFILL(RES.REQ) + RNDS
43 LET ROTY(RES.REQ) = ROTY(RES.REQ) - RNDS
44 ''REDUCE THE ON-HAND BALANCE OF SUPPLY STOCKS
45 LET ONHAND(SCODE(A,RAC(RES.REQ))) =
46 ONHAND(SCODE(A,RAC(RES.REQ))) - RNDS
47 ''REDUCE THE WEIGHT AND CUBE REQ'D FOR THE RES.REQ
48 LET WT.REQ= TRUNC.F(WT.REQ - RNDS
49 * WT.PKG(SCODE(A,RAC(RES.REQ)))
50 /RDS.PKG(SCODE(A,RAC(RES.REQ))))
51 LET CU.REQ= TRUNC.F(CU.REQ - RNDS
52 * CU.PKG(SCODE(A,RAC(RES.REQ)))
53 /RDS.PKG(SCODE(A,RAC(RES.REQ))))
54 ''REDUCE THE WEIGHT AND CUBE AVAIL ON THE TRUCK
55 LET TWT(TANK) = TRUNC.F(TWT(TANK) - RNDS
56 * WT.PKG(SCODE(A,RAC(RES.REQ)))
57 /RDS.PKG(SCODE(A,RAC(RES.REQ))))
58 LET TCU(TANK) = TRUNC.F(TCU(TANK) - RNDS
59 * WT.PKG(SCODE(A,RAC(RES.REQ)))
60 /RDS.PKG(SCODE(A,RAC(RES.REQ))))
61 FILE TANK IN ELEMENTS(CON.ID)
62 LOOP
63 RETURN
64 END

```

EXPLANATION OF CODE

Lines 3-5 Define recursive variables used in the routine.

Line 6 Prints a message indicating that the routine has been called.

Lines 8-11 Begin the major loop for the routine to assign the resupply mission to vehicles in the supply unit that have not previously sustained damage or been assigned to a convoy. Loop ends on line 62.

Lines 12-17 Check the loop for completion through the assignment of all trucks allocated to the mission, or through the completion of the loading process.

Lines 18-32 Check if the request exceeds the carrying capacity of the truck being loaded. If so, the amount being loaded is adjusted to the maximum load for that truck.

Line 33 Creates a TRUCK.CARGO to hold information as to the type and quantity of ammo to be loaded on a truck.

Line 34 Captures the pointer value of the truck the cargo is loaded on.

Line 35 Captures the pointer value of the resupply request being filled.

Line 36 Captures the nomenclature of the request being filled.

Line 37 Captures the ammunition code of the request being filled

Line 38 Captures the quantity being loaded on the truck.

Line 39 Files the cargo on the last truck in the convoy.

Line 41 Reduces the number of rounds yet to be filled for the request

Line 42 Adds the quantity released to the fill attribute of the request.

Line 43 Reduces the required quantity for the resupply request.

Lines 44-46 Reduce the on-hand stocks for the ammo type at the supply point.

Lines 47-53 Reduce the weight and cube required to fill the request.

Lines 54-60 Reduce the weight and cube available on the truck to fill requests.

Line 61 Files the truck in the active convoy.

Line 62 Closes the major routine loop begun on line 8.

Control is either transferred back to fill another truck with the same cargo or transferred out.

M. "ROUTINE WT.AND.CU"

This routine is called by event UP.S4.AMMO and is used to compute the total weight and cube capacity present at battalion which can be used to carry supplies.

ARGUMENT (INTEGER)

A - Points to supply officer updating.

ARGUMENT (REAL)

CU(CUBE) . Holds the maximum cube that may be loaded on a
resupply vehicle.

CU.AVAIL(CUBE AVAILABLE) . Holds the total cube capacity that
is available within the supply unit for the shipment of
supplies.

TRKS.AVAIL(TRUCKS.AVAILABLE) . Holds the total number of
trucks available for assignment to a resupply mission at
any time.

WT(WEIGHT) . Holds the maximum weight that may be loaded on a
resupply vehicle.

WT.AVAIL(WEIGHT AVAILABLE) . Holds the total weight capacity
that is available within the supply unit for the
shipment of supplies.

GLOBAL_VARIABLE_(INTEGER)

TANK

SET

S.UNIT(SUPPLY UNIT) . Contains the unit's supply
vehicles(TANKs) .

TEMPORARY_ATTRIBUTES_(INTEGER)

FKILL(FIREPOWER KILL) .

KKILL(CATASTROPHIC KILL) .

M.ELEMENTS. Specifies membership in a CONVOY.

MAX.CUBE(MAXIMUM CUBE) .

MAX.WT(MAXIMUM WEIGHT).

MFKILL(MOBILITY AND FIREPOWER KILL).

MKILL(MOBILITY KILL).

WT(WEIGHT). Holds the maximum weight that may be loaded on a
resupply vehicle.

ROUTINE_CALLED

WT.AND.CUBE

PROGRAM LISTING

```
1  ROUTINE WT.AND.CUBE GIVEN A
2  YIELDING WT.AVAIL,CU.AVAIL,TRKS.AVAIL,WT,CU
3  DEFINE WT.AVAIL,CU.AVAIL,WT,CU AS REAL VARIABLES
4  DEFINE TRKS.AVAIL,A AS INTEGER VARIABLES
5  PRINT 1 LINE THUS
   ROUTINE WT.AND.CUBE CALLED
6  ''
7  '' DETERMINE THE WT.AND CUBE AVAIL FOR SHIPPING
8  FOR EVERY TANK IN S.UNIT(A)
9  WITH M.ELEMENTS(TANK) EQ 0
10 AND MKILL(TANK) EQ 0 AND FKILL(TANK) EQ 0
11 AND MFKILL(TANK) EQ 0 AND KKILL(TANK) EQ 0, DO
12   LET WT.AVAIL = WT.AVAIL + MAX.WT(TANK)
13   LET CU.AVAIL = CU.AVAIL + MAX.CUBE(TANK)
14   LET TRKS.AVAIL = TRKS.AVAIL + 1
15   LET WT = MAX.WT(TANK)
16   LET CU = MAX.CUBE(TANK)
17 LOOP
18 RETURN
19 END
```

EXPLANATION OF CODE

Lines 3-4 Define recursive variables used in the routine.

Line 5 Prints a message indicating that the routine has
been called.

Lines 8-17 Begin the major loop for the routine, looping
over all available cargo carriers, summing the total
weight and cube available for shipment, determining a

total number of trucks available, and determining the max weight and max cube available on any one truck.

N. "ROUTINE PRI.RESUPPLY"

This routine is called by event UP.S4.AMMO and is used to determine an optimum fill for priority 1 and 2 requisitions. In execution, the routine seeks to fill all priority 2 requisitions to a minimum percentage of fill for all requests by: first, setting multipliers to reduce priority 2 fill; second, re-evaluating the overall fill of priority 2 requests; and third, if necessary, reducing fill on priority 1 requests to open more space for priority 2 requests.

ARGUMENT_(INTEGER)

A - Points to the S-4 currently updating.

LON1. Variable which indicates whether Level of Need 1

requests are currently filed with the supply officer.

LON2. Variable which indicates whether Level of Need 2

requests are currently filed with the supply officer.

ARGUMENT_(REAL)

CU.AVAIL(CUBE AVAILABLE). Holds the total cube capacity that is available within the supply unit for the shipment of supplies.

MUL1(MULTIPLIER 1) . Holds the multiplier for LON 1 requests
which is used to reduce the fill on such requests.

MUL2(MULTIPLIER 2) . Holds the multiplier for LON 2 requests
which is used to reduce the fill on such requests.

WT.AVAIL(WEIGHT AVAILABLE) . Holds the total weight capacity
that is available within the supply unit for the
shipment of supplies.

GLOBAL VARIABLES (INTEGER)

RES.REQ (RESUPPLY REQUEST) .

SCODE(SUPPLY CODE) .

GLOBAL VARIABLES (REAL)

C.L1PCT (CRITICAL LON1 PERCENTAGE) . Holds the maximum
percentage that LON1 requisitions may be reduced to in
order to release space on a convoy for other critical
LON1 and LON2 requests.

C.L2CU (CRITICAL LON2 CUBE) . Holds the maximum percent of
cube that LON2 requisitions may be cut to in order to
release space on a convoy for other critical LON1 and
LON2 requisitions.

C.L2WT (CRITICAL LON2 WEIGHT) . Holds the maximum percent of
weight that LON2 requisitions may be cut to in order to
release space on a convoy for other critical LON1 and
LON2 requisitions.

RECURSIVE_VARIABLE_(REAL)

CU.SHIP(CUBE OF SHIPMENT). Combined total cube of current LON1 and LON2 requests.

C1. Holds the total cube of current LON1 requests.

C1PCT. Percent of LON1 requests which can be filled if all LON2 requests are filled to a minimum based on cube.

C2. Holds the total cube of current LON2 requests.

C2PCT. Percent of LON2 requests which can be filled if all LON1 requests are filled based on cube.

WT.SHIP(WEIGHT OF SHIPMENT). Combined total weight of current LON1 and LON2 requests.

W1. Holds the total weight of current LON1 requests.

W1PCT. Percent of LON1 requests which can be filled if all LON2 requests are filled to a minimum based on weight.

W2. Holds the total weight of current LON2 requests.

W2PCT. Percent of LON2 requests which can be filled if all LON1 requests are filled based on weight.

ROUTINES

MAX.F, MIN.F, PRI.RESUPPLY

SET

SWANT.LIST(SUPPLY WANT LIST).

TEMPORARY_ATTRIBUTE_(INTEGER)

RAC(RESUPPLY AMMUNITION CODE). Attribute of an SCL.V.ITEM
which points to a specific ammo type carried by the
supply unit.

RDS.PKG(ROUNDS PACKAGE). Attribute of a SCL.V.ITEM
specifying the number of rounds on an ammo pallet.

RQTY(RESUPPLY QUANTITY). Attribute of a RES.REQ holding the
total quantity requested by a unit.

SPRIORITY(SUPPLY PRIORITY). Attribute of a RES.REQ
specifying the urgency of need for the ammo request.

TEMPORARY_ATTRIBUTE__(REAL)

CU.PKG(CUBE PACKAGE). Attribute of a SCL.V.ITEM specifying
the cube of an ammunition pallet.

WT.PKG(WEIGHT PACKAGE). Attribute of a SCL.V.ITEM specifying
the weight of an ammunition pallet.

PROGRAM LISTING

```
1  ROUTINE PRI.RESUPPLY GIVEN WT.AVAIL,CU.AVAIL,A
2                                YIELDING MUL1,MUL2,LON1,LON2
3  DEFINE A,LON1,LON2 AS INTEGER VARIABLES
4  DEFINE W1,W2,C1,C2,WT.SHIP,CU.SHIP,MUL1,MUL2,
5  WT.AVAIL,CU.AVAIL,W2PCT,C2PCT,W1PCT,C1PCT
6  AS REAL VARIABLES
7  PRINT 1 LINE THUS
8  ROUTINE PRI.RESUPPLY CALLED
9  '' DETERMINE WEIGHT AND CUBE REQUIRED
10 '' FOR LON1 & LON2 MISSIONS
11 FOR EVERY RES.REQ IN SWANT.LIST(A)
12 WITH SPRIORITY(RES.REQ)=1
13 OR SPRIORITY(RES.REQ)=2, DO
14   GO TO L1,L2 PER SPRIORITY(RES.REQ)
15   'L1' LET W1 = W1 + WT.PKG(SCODE(A,RAC(RES.REQ)))
16         *RQTY(RES.REQ)
17         /RDS.PKG(SCODE(A,RAC(RES.REQ)))
18   LET C1 = C1 + CU.PKG(SCODE(A,RAC(RES.REQ)))
19         *RQTY(RES.REQ)
20         /RDS.PKG(SCODE(A,RAC(RES.REQ)))
21   LET LON1 = 1
22   CYCLE
23   'L2' LET W2 = W2 + WT.PKG(SCODE(A,RAC(RES.REQ)))
```



```

23          *RQTY (RES.REQ)
24          /RDS.PKG (SCODE (A,RAC (RES.REQ)))
25      LET C2 = C2 + WT.PKG (SCODE (A,RAC (RES.REQ)))
26      *RQTY (RES.REQ) /RDS.PKG (SCODE (A,RAC (RES.REQ)))
27      LET LON2 = 1
28  LOOP
29      LET WT.SHIP = WT.SHIP + W1 + W2
30      LET CU.SHIP = CU.SHIP + C2 + C2
31      ''
32      ''DETER IF THE MIN % OF LON 2 ITEMS ARE SHIPPED.
33      IF (WT.SHIP GT WT.AVAIL OR CU.SHIP GT CU.AVAIL)
34      AND LON2 EQ 1
35          LET W2PCT = (WT.AVAIL-W1) /W2
36          LET C2PCT = (CU.AVAIL-C1) /C2
37          ''REDUCE THE FILL ON LON2 REQUIREMENTS
38          LET MUL2 = (MIN.F (W2PCT,C2PCT,1.0) )
39          LET MUL1 = 1.0
40      ''
41      ''REDUCE LON1 FILL TO ALLOW MORE SPACE
42      IF W2PCT LT C.L2WT OR C2PCT LT C.L2CU
43      AND LON1 EQ 1
44          LET W1PCT = (WT.AVAIL-W2*C.L2WT) /W1
45          LET C1PCT = (CU.AVAIL-C2*C.L2CU) /C1
46          LET MUL1 = (MAX.F (W1PCT,C1PCT,C.L1PCT))
47          LET MUL2 = (MIN.F (C.L2WT,C.L2CU,1.0) )
48      ALWAYS
49      ALWAYS
50      RETURN
51      END

```

EXPLANATION OF CODE

Line 3-6 Define recursive variables used in the routine.

Line 7 Prints a message marking the beginning of the routine.

Lines 10-12 Begin the major loop of the routine looping over all resupply requests in the S-4's want list and identify the priority 1 and 2 requests. Loop ends on line 28.

Line 13 Transfers control based on the priority of the request.

Lines 14-16 Sum the total weight of all priority 1 requests.

Lines 17-19 Sum the total cube of all priority 1 requests.

Line 20 Indicates the presence of an LON1 request.

Line 21 Cycles to next request.

Lines 22-24 Sum the total weight of all priority 2 requests.

Lines 25-26 Sum the total cube of all priority 2 requests.

Line 27 Indicates the presence of an LON2 request.

Line 28 Ends the major loop begun on line 10. Transfer of control is either to the next request or out.

Lines 29-30 Calculate the total weight and cube of outstanding requests due to priority 1 and 2 requests.

Lines 32-39 Determine if all priority 2 requests have been filled. If not, it establishes a multiplier, Mul2, to reduce fill on priority 2 requests in order to open up space on trucks to fill a greater number of LON2 requests. The IF check ends on line 49.

Lines 41-48 Check again to see if all LON2 requests have at least been partially filled. If not, it establishes a multiplier, MUL1, to reduce the fill of priority 1 requests to open up space on trucks until the priority 2 requests have been filled. Priority 1 requests will be reduced only to the percentage necessary to fill priority 2 requests, or to a user designated critical priority of fill whichever comes first.

O. "EVENT BAT.L.TIME"

Event BAT.L.TIME is initially scheduled in the main program and subsequently reschedules itself each day of the simulation. The event is used to start and end a battle and to schedule moves.

EVENT_SCHEDULED

MOVE

GLOBAL_VARIABLES_(INTEGER)

CNUM(COMPANY NUMBER). Specifies the number of

COMPANY.COMMANDERS created.

RSTREAM(RESUPPLY RANDOM NUMBER STREAM).

WSTREAM(WEAPON RANDOM NUMBER STREAM).

GLOBAL_VARIABLES_(REAL)

B.END(BATTLE END). Termination time of daily battle.

B.START(BATTLE START). Start time of daily battle.

PERMANENT_ATTRIBUTES_(REAL)

TIME.V

TEMPORARY_ATTRIBUTE_(INTEGER)

MARCH.ORDER. Argument for the event move holding the pointer of the company receiving orders to move.

RECURSIVE_VARIABLES_(INTEGER)

I - Loop index.

ROUTINES_CALLED

RANDOM.F, TRUNC.F, and UNIFORM.F

PROGRAM LISTING

```
1  EVENT BAT.L.TIME
2  ''
3  DEFINE I AS AN INTEGER VARIABLE
4  PRINT 1 LINE WITH TIME.V THUS
   EVENT BAT.L.TIME CALLED AT TIME.V = *.***
5  ''
6  LET B.START = TRUNC.F(TIME.V)
   +UNIFORM.F(0.0,1.00,WSTREAM)
7  LET B.END = B.START +.25
8  IF B.END LT TRUNC.F(TIME.V) + 1.0,
9  RESCHEDULE A BAT.L.TIME AT TRUNC.F(TIME.V) +1.0
10 ELSE
11 RESCHEDULE A BAT.L.TIME AT B.END
12 ALWAYS
13 FOR I = 1 TO CNUM, DO
14 IF RANDOM.F(RSTREAM) LT .8,
15 SCHEDULE A MOVE(I) AT B.END + .1
16 PRINT 3 LINES WITH I THUS
   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
   MOVE SCHEDULED FOR COMPANY *
   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
17 ALWAYS
18 LOOP
19 RETURN
20 END
```

EXPLANATION OF CODE

Line 3 Defines the recursive variable used in the routine.

Line 4 Prints a message marking the start of the routine.

Line 6 Randomly picks a battle start time over the 24 hour period.

Line 7 Schedules a battle stop time 6 hours later.

Lines 8-12 Reschedule the battle for the next day.

Lines 13-18 Randomly select a company to move at the end of a day's battle(event MOVE). Print a message identifying the unit.

P. "EVENT MOVE"

Event MOVE is scheduled in event BAT.L.TIME. The event only takes place after a battle and is used solely to schedule and exercise the redistribution logic.

ARGUMENTS

MARCH.ORDER. Argument for event MOVE holding the pointer of the company receiving orders to move.

GLOBAL_VARIABLE_(INTEGER)

PLATOON.LEADER

SET

CO.UNIT(COMPANY UNIT) .

TEMPORARY_ATTRIBUTES_(INTEGER)

C - Holds the value of the company currently moving.

DISTR(DISTRIBUTOR). Argument for event REDISTRIBUTE holding the value of the unit's PLATOON.LEADER.

EVENT_SCHEDULED

REDISTRIBUTE

PROGRAM_LISTING

```
1  UPON MOVE(C)
2  NORMALLY MODE IS INTEGER
3  FOR EVERY PLATOON.LEADER IN PLAT.UNIT(C), DO
4    SCHEDULE A REDISTRIBUTE(PLATOON.LEADER) NOW
5  LOOP
6  RETURN
7  END
```

EXPLANATION_OF_CODE

Line 2 Defines the mode as integer.

Lines 3-5 Schedule a move for each platoon in the company.

Q. "EVENT CO.RESUPPLY.ARR"

Event CO.RESUPPLY.ARR is scheduled from event UP.S4.AMMO. The event distributes the ammunition assets received by a company to subordinate platoons according to their most recent LON. It then sends the RSV/convoy back to the battalion trains after an appropriate time delay simulating a delivery from the ATP/ASP.

ARGUMENT

CNVY(CONVOY). Argument of CO.RESUPPLY.ARR(COMPANY RESUPPLY ARRIVE) carrying the pointer of the convoy arriving.

CO.CNVY(COMPANY CONVOY). Argument of the event

CO.RESUPPLY.ARR (COMPANY RESUPPLY ARRIVE) holding the pointer value for the convoy arriving in the area.

DEFINE TO MEAN

AMMO1. AP.TOW(ARMOR PIERCING/TOW ROUNDS).

AMMO2. HE.DRAG(HEAT/DRAGON ROUNDS).

AMMO3. AW1.OR.MSL3(ALTERNATE WEAPON 1 OR MISSILE 3).

AMMO4. AW2.OR.ADM(ALTERNATE WEAPON 2 OR AIR DEFENSE MISSILE).

EVENTS SCHEDULED

BN.ARRIVE(BATTALION ARRIVE).

GLOBAL VARIABLES (INTEGER)

PCODE(PLATOON CODE) (2-d) . Holds the pointer value for a
platoon's PCL.V.ITEMS(PLATOON CLASS V ITEMS) .

PLATOON.LEADER

RES.REQ (RESUPPLY REQUEST) .

TANK

TSTREAM (TRIP RANDOM NUMBER STREAM) .

GLOBAL_VARIABLES_(REAL)

CCODE(COMPANY AMMUNITION CODE) .

MAXTRIP(MAX TRIP) . The maximum time required for a convoy to
reach its intended destination.

MINTRIP(MIN TRIP) . The minimum time required for a convoy to
reach its intended destination.

PERMANENT_ATTRIBUTES_(REAL)

TIME.V

RECURSIVE_VARIABLES_(INTEGER)

ASSETS. Holds the amount of ammo left to be issued.

DEL. Holds the amount of ammo initially delivered.

NO.BATTLE. Indicates whether RES.REQs should be filled.

"0" indicates no

"1" indicates yes

TEMP. Place holder for release point of the convoy.

ROUTINES_CALLED

COM.AMMO, INT.F, P.CLASS.V

SETS_USED

C.CGO.LIST(CONVOY CARGO LIST).

CO.UNIT(COMPANY UNIT).

CWANT.LIST(COMPANY WANT LIST).

PLAT.UNIT(PLATOON UNIT).

SREQN.LIST(SUPPLY REQUISITION LIST).

SWANT.LIST(SUPPLY WANT LIST).

TNK.ALIVE(TANK ALIVE).

TEMPORARY_ATTRIBUTES (INTEGER)

AMMO5(AMMUNITION 5). Of TANK.

AMMO6(AMMUNITION 6). Of TANK.

AP.TOW(ARMOR-PIERCING/TOW). Ammunition 1 OF TANK.

TAC1.(TANK AMMUNITION CODE 1).

TAC2.(TANK AMMUNITION CODE 2).

TAC3.(TANK AMMUNITION CODE 3).

TAC4.(TANK AMMUNITION CODE 4).

TAC5.(TANK AMMUNITION CODE 5).

TAC6.(TANK AMMUNITION CODE 6).

AW1.OR.MSL3(ALTERNATE WEAPON 1 OR MISSILE 3). Ammunition 3.

AW2.OR.ADM(ALTERNATE WEAPON 2 OR AIR DEFENSE MISSILE).

Ammunition 4.

C.MV.STATE(CONVOY MOVEMENT STATE).

C.SHORT (COMPANY SHORTAGE). Total rounds short for a type
ammo.

COCDR (COMPANY COMMANDER). Of TANK.

FKILL (FIREPOWER KILL).

FLAG Yielding argument of W.AMMO and COM.AMMO, not used in
this routine.

HE.DRAG (HEAT/DRAGON ROUNDS). Ammunition 2 of TANK.

KKILL (CATASTROPHIC KILL).

MKILL (MOBILITY KILL).

OH1 (ON-HAND 1). Current balance of ammunition 1 on a TANK.

OH2 (ON-HAND 2). Current balance of ammunition 2 on a TANK.

OH3 (ON-HAND 3). Current balance of ammunition 3 on a TANK.

OH4 (ON-HAND 4). Current balance of ammunition 4 on a TANK.

OH5 (ON-HAND 5). Current balance of ammunition 5 on a TANK.

OH6 (ON-HAND 6). Current balance of ammunition 6 on a TANK.

P.SHORT (PLATOON SHORTAGE). Current shortage of a PCL.V.ITEM
(PLATOON CLASS V ITEM). Unique to each platoon and ammo
type.

PCURR.LOAD (PLATOON CURRENT LOAD). On-hand balance for an
ammo type. Unique for each platoon and ammo type.

RAC (RESUPPLY AMMUNITION CODE).

RCNVY(RESUPPLY CONVOY). Argument of the event BN.ARRIVE

(BATTALION ARRIVE) carrying the pointer of the convoy.

REQUESTOR. Attribute of a RES.REQ(RESUPPLY REQUEST)

specifying the unit making the request.

RFILL(RESUPPLY FILL). Attribute of a RES.REQ (RESUPPLY

REQUEST) specifying the amount of ammunition released to
fill a request.

RP(RELEASE POINT). Attribute of a convoy specifying the

convoy's destination.

SLOAD1(STOWED LOAD 1). Optimal load ammo type 1.

SLOAD2(STOWED LOAD 2). Optimal load ammo type 2.

SLOAD3(STOWED LOAD 3). Optimal load ammo type 3.

SLOAD4(STOWED LOAD 4). Optimal load ammo type 4.

SLOAD5(STOWED LOAD 5). Optimal load ammo type 5.

SLOAD6(STOWED LOAD 6). Optimal load ammo type 6.

SP(START POINT). Attribute of a convoy specifying the

convoy's origin.

STATUS. Indicates the current location of a convoy.

SYS.TYPE(SYSTEM TYPE).

ROUTINES CALLED

UNIFORM.F, UP.DATE, W.AMMO

PROGRAM LISTING

```
1  EVENT CO.RESUPPLY.ARR(CNVY)
2  DEFINE FLAG,NO.BATTLE,TEMP,CNVY,ASSETS,
```



```

3  AND DEL AS INTEGER VARIABLES
4  PRINT 2 LINES WITH TIME.V,CNVY THUS
   EVENT CO.RESUPPLY.ARR CALLED
   TIME.V = **.****, CONVOY = ***
5  ''
6  ''BRING THE CONVOY OUT OF HYPERSPACE
7  LET C.MV.STATE(CNVY) = 0
8  ''
9  FOR EVERY RES.REQ IN C.CGO.LIST(CNVY), DO
10 LET ASSETS = Rfill(RES.REQ)
11 ''
12 FOR EVERY PLATOON.LEADER
13   IN CO.UNIT(REQUESTOR(RES.REQ)), DO
14   LET DEL=
15   INT.F(P.SHORT(PCODE(PLATOON.LEADER,RAC(RES.REQ)))
16   /C.SHORT(CCODE(REQUESTOR(RES.REQ),RAC(RES.REQ)))
17   * Rfill(RES.REQ))
18   LET ASSETS = ASSETS - DEL
19   LET PCURR.LOAD(PCODE(PLATOON.LEADER,RAC(RES.REQ))) =
20   PCURR.LOAD(PCODE(PLATOON.LEADER,RAC(RES.REQ))) + DEL
21 ''
22   ''FILL TANKS
23   FOR EVERY TANK IN PLAT.UNIT(PLATOON.LEADER), DO
24     IF MKILL(TANK)=1 OR MFKILL(TANK)=1
25     OR FKILL(TANK)=1 OR KILL(TANK)=1
26     CYCLE
27     OTHERWISE
28 ''
29     IF TAC1(TANK)=RAC(RES.REQ)
30     LET OH1(TANK) = OH1(TANK) + DEL
31     ( (SLOAD1(TANK)-OH1(TANK))) /
32     P.SHORT(PCODE(PLATOON.LEADER,RAC(RES.REQ)))
33     LET AMMO1(TANK)=AMMO1(TANK)+DEL
34     ( (SLOAD1(TANK)-AMMO1(TANK))) /
35     P.SHORT(PCODE(PLATOON.LEADER,RAC(RES.REQ)))
36     GO TO TANKLOOP
37     OTHERWISE
38 ''
39     IF TAC2(TANK)=RAC(RES.REQ)
40     LET OH2(TANK)=OH2(TANK)+DEL
41     ( (SLOAD2(TANK)-OH2(TANK))) /
42     P.SHORT(PCODE(PLATOON.LEADER,RAC(RES.REQ)))
43     LET AMMO2(TANK)=AMMO2(TANK)+DEL
44     ( (SLOAD2(TANK)-AMMO2(TANK))) /
45     P.SHORT(PCODE(PLATOON.LEADER,RAC(RES.REQ)))
46     GO TO TANKLOOP
47     OTHERWISE
48 ''
49     IF TAC3(TANK)=RAC(RES.REQ)
50     LET OH3(TANK) = OH3(TANK) + DEL
51     ( (SLOAD3(TANK)-OH3(TANK))) /
52     P.SHORT(PCODE(PLATOON.LEADER,RAC(RES.REQ)))
53     LET AMMO3(TANK)=AMMO3(TANK)+DEL
54     ( (SLOAD3(TANK)-AMMO3(TANK))) /
55     P.SHORT(PCODE(PLATOON.LEADER,RAC(RES.REQ)))
56     GO TO TANKLOOP
57     OTHERWISE
58 ''
59     IF TAC4(TANK)=RAC(RES.REQ)
60     LET OH4(TANK) = OH4(TANK) + DEL
61     ( (SLOAD4(TANK)-OH4(TANK))) /
62     P.SHORT(PCODE(PLATOON.LEADER,RAC(RES.REQ)))
63     LET AMMO4(TANK)=AMMO4(TANK)+DEL
64     ( (SLOAD4(TANK)-AMMO4(TANK))) /
65     P.SHORT(PCODE(PLATOON.LEADER,RAC(RES.REQ)))
66     GO TO TANKLOOP
67     OTHERWISE
68 ''

```



```

69      IF TAC5 (TANK) = RAC (RES.REQ)
70      LET OH5 (TANK) = OH5 (TANK) + DEL
71      ( (SLOAD5 (TANK) - OH5 (TANK)) ) /
72      P.SHORT (PCODE (PLATOON.LEADER, RAC (RES.REQ)) )
73      LET AMMO5 (TANK) = AMMO5 (TANK) + DEL
74      ( (SLOAD5 (TANK) - AMMO5 (TANK)) ) /
75      P.SHORT (PCODE (PLATOON.LEADER, RAC (RES.REQ)) )
76      GO TO TANKLOOP
77  OTHERWISE
78  ''
79      IF TAC6 (TANK) = RAC (RES.REQ)
80      LET OH6 (TANK) = OH6 (TANK) + DEL
81      ( (SLOAD6 (TANK) - OH6 (TANK)) ) /
82      P.SHORT (PCODE (PLATOON.LEADER, RAC (RES.REQ)) )
83      LET AMMO6 (TANK) = AMMO6 (TANK) + DEL
84      ( (SLOAD6 (TANK) - AMMO6 (TANK)) ) /
85      P.SHORT (PCODE (PLATOON.LEADER, RAC (RES.REQ)) )
86  ALWAYS
87  ''
88      'TANKLOOP'
89  LOOP
90      'UPDATEFILES'
91      IF RES.REQ IS NOT IN SOME SWANT.LIST
92      AND RES.REQ IS IN SOME CWANT.LIST,
93      REMOVE RES.REQ
94      FROM CWANT.LIST (REQUESTOR (RES.REQ) )
95  ALWAYS
96  ''
97      IF RES.REQ IS NOT IN SOME SREQN.LIST
98      FILE RES.REQ IN SREQN.LIST (SP (CNVY) )
99  ALWAYS
100     LET STATUS (RES.REQ) = "ATP"
101     LET DEL = 0
102  LOOP
103  LOOP
104  FOR EVERY TANK IN TNK.ALIVE
105  WITH COCDR (TANK) EQ RP (CNVY) , DO
106  IF SYS.TYPE (TANK) EQ 7,
107  CYCLE
108  OTHERWISE
109  LET NO.BATTLE = 1
110  CALL W.AMMO (TANK, NO.BATTLE) YIELDING FLAG
111  LOOP
112  ''
113  FOR EVERY PLATOON.LEADER IN CO.UNIT (RP (CNVY)) , DO
114  CALL P.CLASS.V (PLATOON.LEADER)
115  LOOP
116  LET NO.BATTLE = 1
117  CALL COM.AMMO (RP (CNVY) , NO.BATTLE) YIELDING FLAG
118  ''
119  'SEND THE CONVOY HOME
120  LET TEMP = RP (CNVY)
121  LET RP (CNVY) = SP (CNVY)
122  LET SP (CNVY) = TEMP
123  LET C.MV.STATE (CNVY) = 1
124  SCHEDULE A BN.ARRIVE (CNVY)
125  IN UNIFORM.F (MINTRIP, MAXTRIP, TSTREAM) MINUTES
126  RETURN
127  END
128

```

EXPLANATION OF CODE

Lines 2-3 Define the recursive variables used in the routine.

Line 4 Prints a message marking the start of the routine.

Lines 6-7 Change the convoy movement state to zero.

Line 9 Begins the routine's major loop over all resupply requests in the arriving convoys cargo list. Loop ends on line 104.

Line 10 Captures the quantity delivered by the convoy as a local variable for computation purposes.

Lines 12-13 Begin an inner loop over every platoon leader in the company in order to distribute the arriving supplies. Loop ends on line 103.

Lines 14-17 Set the delivery for a particular platoon equal to $[\text{platoon shortage} / \text{company shortage}]$ multiplied by the deliver quantity.

Line 18 Subtracts the delivery from the assets available.

Lines 19-20 Adjust the platoon current load for that ammo type.

Line 22 Begins an inner loop over all the weapon systems in the platoon so that deliveries can be made. Loop ends on line 89.

Lines 24-27 Determine if any vehicle has sustained casualties, if so, the loop is cycled to the next weapon system.

Lines 29-86 Check to see if the ammo delivered is one of the six carried on the TANK. If so, the weapon system's on-hand knowledge and actual knowledge is updated to reflect the delivery.

Lines 88-89 Close the combat vehicle loop begun on line 22. Transfer of control goes to the next TANK or out to the next platoon.

Lines 91-101 Update files by removing the request from company want lists and filing the same request on a supply request, simulating a request from battalion to the brigade ATP for more ammo.

Line 102 Resets the delivery quantity to zero.

Line 103 Closes out the platoon loop begun on line 12.

Transfer of control is to the next platoon or out to the next request.

Line 104 Closes out the request loop begun on line 9.

Transfer of control is to the next request or out.

Lines 105-112 Cause every TANK in the company to update its ammo status.

Lines 114-116 Cause every platoon in the company to update its ammo status.

Lines 117-118 Cause the company to update its ammo status.

Lines 120-126 Turn the convoy around and sends it back to the battalion supply point as a resupply coming from the ATP.

R. "EVENT REDISTRIBUTE"

This event is scheduled from event MOVE. It is used to evenly redistribute ammunition in a platoon.

ARGUMENTS (INTEGER)

P - Holds the pointer value of the platoon currently redistributing.

DISTR(DISTRIBUTOR). Argument for the event REDISTRIBUTE holding the value of the unit's PLATOON.LEADER.

DEFINE TO MEAN

AMMO1. AP.TOW (ARMOR PIERCING/TOW ROUNDS) .

AMMO2. HE.DRAG (HEAT/DRAGON ROUNDS) .

AMMO3. AW1.OR.MSL3 (ALTERNATE WEAPON 1 OR MISSILE 3) .

AMMO4. AW2.OR.ADM (ALTERNATE WEAPON 2 OR AIR DEFENSE MISSILE) .

GLOBAL VARIABLES (INTEGER)

PCL.V.ITEM

PCODE(PLATOON CODE) (2-d) . Holds the pointer value for a
platoon's PCL.V.ITEMS(PLATOON CLASS V ITEMS) .

TANK

PERMANENT_ATTRIBUTE (REAL)

TIME.V

RECURSIVE_VARIABLE (INTEGER)

CASSETS. Current amount of an ammo type on-hand in a
platoon.

FLAG. Yielding argument for routine W.AMMO; not used in this
routine.

NO.BATTLE. Indicates whether RES.REQs should be filled.

"0" indicates no

"1" indicates yes

RASSETS. Required amount of an ammo type based on the
 platoons stowed load.

ROUTINES_CALLED

P.CLASS.V, UP.DATE, W.AMMO

SETS_USED

PLAT.UNIT(PLATOON UNIT). Owned by a PLATOON.LEADER. Members
are the unit's combat vehicles(TANKs) .

PLT.AMMO(PLATOON AMMUNITION). Owned by a PLATOON.LEADER.
Members are the unit's PCL.V.ITEMS(PLATOON CLASS V
ITEMS) .

TEMPORARY_ATTRIBUTE (INTEGER) .

AMMO5 (AMMUNITION 5) .

AMMO6 (AMMUNITION 6) .

AP.TOW (ARMOR-PIERCING/TOW) . Ammunition 1

TAC1. (TANK AMMUNITION CODE 1) .

TAC2. (TANK AMMUNITION CODE 2) .

TAC3. (TANK AMMUNITION CODE 3) .

TAC4. (TANK AMMUNITION CODE 4) .

TAC5. (TANK AMMUNITION CODE 5) .

TAC6. (TANK AMMUNITION CODE 6) .

AW1.OR.MSL3 (ALTERNATE WEAPON 1 OR MISSILE 3) . Ammunition 3.

AW2.OR.ADM (ALTERNATE WEAPON 2 OR AIR DEFENSE MISSILE) .

Ammunition 4.

FKILL (FIREPOWER KILL) .

HE.DRAG (HEAT/DRAGON ROUNDS) . Ammunition 2.

KKILL (CATASTROPHIC KILL) .

MFKILL (MOBILITY AND FIREPOWER KILL) .

MKILL (MOBILITY KILL) .

PAC (COMPANY AMMUNITION CODE) .

PCURR.LOAD (PLATOON CURRENT LOAD) . On-hand balance for an
ammo type.

PL.B.LOAD (PLATOON BASIC LOAD) .

SLOAD1 (STOWED LOAD 1) . Optimal load ammo type 1.

SLOAD2 (STOWED LOAD 2) . Optimal load ammo type 2.

SLOAD3(STOWED LOAD 3) . Optimal load ammo type 3.

SLOAD4(STOWED LOAD 4) . Optimal load ammo type 4.

SLOAD5(STOWED LOAD 5) . Optimal load ammo type 5.

SLOAD6(STOWED LOAD 6) . Optimal load ammo type 6.

PROGRAM LISTING

```
1  EVENT REDISTRIBUTE(P)
2  DEFINE NO.BATTLE,RASSETS,CASSETS,ROUND,
3  FLAG,AND P AS INTEGER VARIABLES
4  PRINT 1 LINE WITH TIME.V AS FOLLOWS
   EVENT REDISTRIBUTE CALLED AT TIME.V = **.*
5  ''
6  FOR EVERY TANK IN PLAT.UNIT(P),DO
7    LET NO.BATTLE = 1
8    CALL W.AMMO(TANK,NO.BATTLE) YIELDING FLAG
9  LOOP
10 ''
11 CALL P.CLASS.V(P)
12 FOR EVERY PCL.V.ITEM IN PLT.AMMO(P), DO
13   LET CASSETS = PCURR.LOAD(PCODE(P,ROUND))
14   LET RASSETS = PL.B.LOAD(PCODE(P,ROUND))
15 ''
16   FOR EVERY TANK IN PLAT.UNIT(P),DO
17     IF MKILL(TANK)=1 OR MFKILL(TANK)=1
18     OR FKILL(TANK)=1 OR KILL(TANK)=1
19     CYCLE
20   OTHERWISE
21     IF TAC1(TANK) = PAC(PCL.V.ITEM)
22       LET AMMO1(TANK)=SLOAD1(TANK)/RASSETS*CASSETS
23     CYCLE
24   ALWAYS
25     IF TAC2(TANK) = PAC(PCL.V.ITEM)
26       LET AMMO2(TANK) = SLOAD1(TANK)/RASSETS*CASSETS
27     CYCLE
28   ALWAYS
29     IF TAC1(TANK) = PAC(PCL.V.ITEM)
30       LET AMMO1(TANK) = SLOAD1(TANK)/RASSETS*CASSETS
31     CYCLE
32   ALWAYS
33     IF TAC2(TANK) = PAC(PCL.V.ITEM)
34       LET AMMO2(TANK) = SLOAD2(TANK)/RASSETS*CASSETS
35     CYCLE
36   ALWAYS
37     IF TAC3(TANK) = PAC(PCL.V.ITEM)
38       LET AMMO3(TANK) = SLOAD3(TANK)/RASSETS*CASSETS
39     CYCLE
40   ALWAYS
41     IF TAC4(TANK) = PAC(PCL.V.ITEM)
42       LET AMMO4(TANK) = SLOAD4(TANK)/RASSETS*CASSETS
43     CYCLE
44   ALWAYS
45     IF TAC5(TANK) = PAC(PCL.V.ITEM)
46       LET AMMO5(TANK) = SLOAD5(TANK)/RASSETS*CASSETS
47     CYCLE
48   ALWAYS
49     IF TAC6(TANK) = PAC(PCL.V.ITEM)
50       LET AMMO6(TANK) = SLOAD6(TANK)/RASSETS*CASSETS
51     CYCLE
```



```

52         ALWAYS
53     LOOP
54 LOOP
55 FOR EVERY TANK IN PLAT.UNIT(P) , DO
56     LET NO.BATTLE = 1
57     CALL W.AMMO (TANK,NO.BATTLE) YIELDING FLAG
58 LOOP
59 CALL P.CLASS.V
60 RETURN
61 END

```

EXPLANATION OF CODE

Lines 2-3 Define recursive variables for use in the routine.

Line 4 Prints a message marking the beginning of the routine.

Lines 6-9 Update the platoon weapon system's current knowledge of its ammo status.

Line 11 Updates the platoon's current ammo knowledge.

Line 12 Begins the major loop for the routine by looping over all ammo types carried by the platoon. Loop ends on line 54.

Lines 13-14 Capture the current assets on-hand and the required assets needed to be on-hand for computations.

Line 16 Begins an inner loop for the routine looping over all weapon systems in the platoon. Loop ends on line 53.

Line 17-20 Check if a weapon has sustained battle damage. If so, the routine cycles to the next weapon.

Lines 21-52 Check if the ammo being considered is one of the weapons six ammunitions carried. If so, the weapon is given a percentage of the assets on-hand equal to [weapon stowed load / platoon base load] multiplied by the platoon's assets.

Line 53 Closes out the inner loop causing the routine to loop to line 16 and the next weapon or out to the next ammo.

Line 54 Closes out the major loop causing the routine to loop to line 12 and the next ammo type or out.

Lines 55-58 Cause every TANK in the platoon to update its ammo status.

Line 59 Causes the platoon to update its ammo status.

S. "EVENT BN.ARRIVE"

Event BN.ARRIVE is scheduled from CO.RESUPPLY.ARR to simulate an RSV or convoy returning to the battalion trains after a resupply mission. The RSV or convoy returns to the battalion trains with an identical cargo to what it delivered to the resupplied company. Understandably, this is unrealistic but it serves the purpose of resupplying the battalion trains in the model. This cargo is now added to the battalion's assets and becomes available for resupply again.

ARGUMENTS

RCNVY(RESUPPLY CONVOY). Argument of the event BN.ARRIVE
(BATTALION ARRIVE) carrying the pointer of the convoy.

EVENTS_SCHEDULED

BN.ARRIVE(BATTALION ARRIVE).

GLOBAL_VARIABLES_(INTEGER)

SCODE(SUPPLY CODE) (2-d). Holds the pointer value for a
supply unit's SCL.V.ITEMS(SUPPLY CLASS V ITEMS).

TANK

PERMANENT_ATTRIBUTES

TIME.V

SETS

CARGO. Of a resupply truck.

C.CGO.LIST(CONVOY CARGO LIST). Owned by a CONVOY. Members
are the RES.REQS(RESUPPLY REQUEST) being shipped.

ELEMENTS. Owned by a CONVOY. Members are trucks(TANKs)
belonging to the convoy.

SCONVOY(SUPPLY CONVOY). Owned by a SUPPLY.OFFICER. Members
are CONVOY(s).

SREQN.LIST(SUPPLY REQUISITION LIST). Of a RES.REQ.

TEMPORARY_ENTITIES

CONVOY

RES.REQ. (RESUPPLY REQUEST).

T.CGO (TRUCK CARGO) .

TEMPORARY_ATTRIBUTES_(INTEGER)

C.MV.STATE (COMPANY MOVEMENT STATE) .

ONHAND

RAC (RESUPPLY AMMO CODE) .

RFILL (RESUPPLY FILL) . Attribute of a RES.REQ (RESUPPLY REQUEST) specifying the amount of ammunition released to fill a request.

RP (RELEASE POINT) . Attribute of a CONVOY specifying the convoy's destination.

RQTY (REQUIRED QUANTITY) . Of a RES.REQ.

ROUTINES_CALLED

UP.DATE

PROGRAM_LISTING

```
1  EVENT BN.ARRIVE(RCNVY)
2  DEFINE RCNVY AS AN INTEGER VARIABLE
3  PRINT 1 LINE WITH TIME.V THUS
   EVENT BN.ARRIVE CALLED AT TIME.V = **.*
4  LET C.MV.STATE(RCNVY) = 0
5  FOR EVERY RES.REQ IN C.CGO.LIST(RCNVY), DO
6    ADD RFILL(RES.REQ) TO
7    ONHAND(SCODE(RP(RCNVY), RAC(RES.REQ)))
8    REMOVE RES.REQ FROM C.CGO.LIST(RCNVY)
9    REMOVE RES.REQ FROM SREQN.LIST(RP(RCNVY))
10   IF RQTY(RES.REQ) = 0,
11     DESTROY RES.REQ
12   ALWAYS
13   LOOP
14   FOR EVERY TANK IN ELEMENTS(RCNVY), DO
15     FOR EVERY T.CGO IN CARGO(TANK), DO
16       REMOVE T.CGO FROM CARGO(TANK)
17       DESTROY T.CGO
18     LOOP
19     REMOVE TANK FROM ELEMENTS(RCNVY)
20   LOOP
21   REMOVE RCNVY FROM SCONVOY(RP(RCNVY))
22   DESTROY CONVOY CALLED RCNVY
23   RETURN
24   END
```


EXPLANATION OF CODE

Line 2 Defines recursive variables used in the routine.

Line 3 Prints a message indicating that the routine has been called.

Line 4 Changes the convoy move state to zero.

Lines 5-13 Loop over every resupply request in the convoy adding its fill to the on-hand stocks at the battalion, removing it from the convoy cargo list and the S-4's request list. If the out standing balance for the request is zero it is destroyed.

Lines 14-20 Loop over every TANK in the convoy's elements removing the cargo entities from the trucks, and the trucks from the convoys. The cargo entity is then destroyed.

Lines 21-22 Remove the convoy from the S-4's list and destroys the convoy.

T. "EVENT UP.W.AMMO"

Event UP.W.AMMO is initially scheduled in routine BLU.CREATE for each weapon system in the model. It is used to call routine W.AMMO for a periodic update of ammo LONs. It subsequently randomly reschedules itself simulating the

irregular counting of ammunition during combat. Based on a returning argument from routine W.AMMO, this event may schedule a platoon update if the weapon is critical for an ammo type, or may stop scheduling updates for the weapon if damage has been sustained.

ARGUMENT (INTEGER)

RND.CNTR. Points to weapon currently updating.

GLOBAL VARIABLE (INTEGER)

WSTREAM (WEAPON RANDOM NUMBER STREAM).

GLOBAL VARIABLES (REAL)

WMAX (WEAPON MAXIMUM). The maximum time that can pass before a weapon will update its ammunition status.

WMIN (WEAPON MINIMUM). The minimum time that can pass before a weapon will update its ammunition status.

RECURSIVE VARIABLES (INTEGER)

FLAG. Yielding argument from W.AMMO.

ROUTINES CALLED

UNIFORM.F, UP.PLT.AMMO (UPDATE PLATOON AMMO)

PERMANENT ATTRIBUTE (REAL)

TIME.V

TEMPORARY ATTRIBUTE (INTEGER)

A - Holds pointer of weapon currently updating.

PLTLDR. Of TANK.

P.RND.CNTR (PLATOON ROUND COUNTER). Argument of event

UP.PLT.AMMO pointing to a specific platoon which will
update.

PROGRAM LISTING

```
1  EVENT UP.W.AMMO(A)
2  DEFINE NO.BATTLE,A, AND FLAG AS INTEGER VARIABLES
3  LET NO.BATTLE = 0
4  CALL W.AMMO GIVEN A,NO.BATTLE YIELDING FLAG
5  IF FLAG = 100, '' EMERGENCY RESUPPLY NEEDED
6  SCHEDULE AN'UP.PLT.AMMO(PLTLDR(A)) NOW
7  PRINT 2 LINES WITH TIME.V THUS
   UP.PLT.AMMO CALLED BECAUSE OF ZERO BAL
   TIME.V = **,****
8  ALWAYS
9  IF FLAG NE 1, '' NO BATTLE DAMAGE SUSTAINED
10 SCHEDULE AN UP.W.AMMO(A)
11 IN UNIFORM.F(WMIN,WMAX,WSTREAM) HOURS
12 ALWAYS
13 RETURN
14 END
```

EXPLANATION OF CODE

Line 2 Defines recursive variables for the routine.

Line 3-4 Set the index necessary to call for a battle
update and calls routine W.AMMO.

Lines 5-8 If the flag returned indicates that a weapon is
at empty for an ammunition type, this provokes a call
for the platoon to update its ammo.

Lines 9-11 If the flag indicates that no battle damage has
been sustained, the update is rescheduled.

U. "EVENT UP.PLT.AMMO"

This event is initially scheduled from routine

BLU.CREATE to call routine P.CLASS.V for a periodic ammo

update. It then randomly reschedules itself in order to simulate a platoon leader checking individual weapons. Scheduling also occurs if a weapon reaches an LON of "1" for an ammunition type simulating the initiation of an emergency request. Additionally, based on a return argument from routine P.CLASS.V, this event may schedule a company update if the platoon is critical for an ammo type, or may stop scheduling updates for the platoon if all its weapons are incapacitated.

ARGUMENT_(INTEGER)

P.RND.CNTR (PLATOON ROUND COUNTER). Argument of the event UP.PLT.AMMO (UPDATE PLATOON AMMUNITION) carrying the pointer value of the platoon currently updating.

SET

PLT.AMMO (PLATOON AMMUNITION). Owned by a PLATOON.LEADER.

Members are the unit's PCL.V.ITEMS (PLATOON CLASS V ITEMS).

EVENT_NOTICE

UP.COM.AMMO (UPDATE COMPANY AMMUNITION).

GLOBAL_VARIABLE_(INTEGER)

PSTREAM (PLATOON RANDOM NUMBER STREAM).

GLOBAL_VARIABLES_(REAL)

PMAX(PLATOON MAXIMUM) . The maximum time a that can pass
before a platoon will update its ammunition status.

PMIN(PLATOON MINIMUM) . The minimum time a that can pass
before a platoon will update its ammunition status.

PERMANENT_ATTRIBUTES_(INTEGER)

PCO.CDR(PLATOON COMPANY COMMANDER) .

N.PCL.V.ITEMS (NUMBER OF PLATOON CLASS V ITEMS) .

PERMANENT_ATTRIBUTE_(REAL)

TIME.V

RECURSIVE_VARIABLE_(INTEGER)

FLAG. Yielding argument from P.CLASS.V.

ROUTINES

P.CLASS.V(PLATOON CLASS V AMMO), UNIFORM.F

TEMPORARY_ATTRIBUTE_(INTEGER)

C.RND.CNTR (COMPANY ROUND COUNTER) . Argument of the event

UP.COM.AMMO (UPDATE COMPANY AMMUNITION) carrying the
pointer value of the company currently updating.

J - Points to platoon currently updating.

PROGRAM_LISTING

```
1  EVENT UP.PLT.AMMO(J)
2  DEFINE J, FLAG AS INTEGER VARIABLES
3  CALL P.CLASS.V GIVEN J YIELDING FLAG
4  IF FLAG = 1,
5      SCHEDULE AN UP.COM.AMMO(PCO.CDR(J)) NOW
6      PRINT 2 LINES WITH TIME.V THUS
          UP.COM.AMMO CALLED BECAUSE OF ZERO BAL
          TIME.V = **.****
7  ALWAYS
8  IF FLAG NE N.PCL.V.ITEMS(J), ''PLATOON STILL VIABLE
9      SCHEDULE AN UP.PLT.AMMO(J)
```



```
10     IN UNIFORM.F(PMAX,PMIN,PSTREAM) HOURS
11     ALWAYS
12     RETURN
13     END
```

EXPLANATION OF CODE

Line 2 Defines the recursive variables used in the routine.

Line 3 Calls upon routine P.CLASS.V to update the platoon's ammo status.

Lines 4-7 If the flag equals 1, this indicates that the platoon is at zero balance for an ammo type and the company is requested to update its ammo status.

Lines 8-11 If the flag does not indicate that the platoon has sustained enough damage to place it out of combat, its next update is established.

V. "EVENT UP.COM.AMMO"

Event UP.COM.AMMO is initially scheduled from routine BASIC.LOAD and subsequently randomly reschedules itself simulating the update of ammunition assets within a company. It is also called immediately if a platoon reaches an LON of "1" for an ammunition type simulating the initiation of an emergency request. Additionally, based on a returned argument from routine COM.AMMO, this event may stop scheduling updates if all its platoons have been put hors de combat.

ARGUMENTS

C.RND.CNTR (COMPANY ROUND COUNTER). Argument for event
UP.COM.AMMO (UPDATE COMPANY AMMUNITION) holding a
pointer of a company unit.

EVENTS_SCHEDULED

UP.COM.AMMO

GLOBAL_VARIABLES_(REAL)

CMAX (COMPANY MAXIMUM). The maximum time that can pass before
a company will update its ammunition status.

CMIN (COMPANY MINIMUM). The minimum time that can pass before
a company will update its ammunition status.

CSTREAM (COMPANY RANDOM NUMBER STREAM).

PERMANENT_ATTRIBUTES_(INTEGER)

N.CCL.V.ITEMS (NUMBER COMPANY CLASS V ITEMS).

RECURSIVE_VARIABLES_(INTEGER)

C - Points to company updating.

FLAG. Yielding argument of routine COM.AMMO.

NO.BATTLE. Indicates whether RES.REQs should be filled.

"0" indicates no

"1" indicates yes

ROUTINES_CALLED

COM.AMMO (COMPANY AMMO), UNIFORM.F

PROGRAM_LISTING

```
1  EVENT UP.COM.AMMO(C)
2  DEFINE NO.BATTLE,C,FLAG AS INTEGER VARIABLES
3  LET NO.BATTLE = 0
4  CALL COM.AMMO GIVEN C,NO.BATTLE  YIELDING FLAG
```



```

5  IF FLAG NE N.CCL.V.ITEMS (C) ,
6    SCHEDULE A UP.COM.AMMO (C)
7    IN UNIFORM.F (CMIN,CMAX,CSTREAM)  HOURS
8  ALWAYS
9  RETURN
10 END

```

EXPLANATION OF CODE

Line 2 Defines the recursive variables used in the routine.

Lines 3-4 Set the indicator flag to require requisitions to be filed and calls routine COM.AMMO.

Lines 5-8 If the indicator flag returned indicates that the company is still a viable combat entity, its next update is scheduled.

W. "EVENT B.UP.DATE"

Event B.UP.DATE is initially scheduled in the main program to initiate a periodic call for a battle summary from routine UP.DATE. This event subsequently reschedules itself every 24 hours.

PERMANENT ATTRIBUTE

TIME.V

ROUTINES CALLED

TRUNC.F, UP.DATE

PROGRAM LISTING

```

1  EVENT B.UP.DATE
2  CALL UP.DATE
3  SCHEDULE A B.UP.DATE AT TRUNC.F (TIME.V) + 1
4  RETURN
5  END

```

EXPLANATION OF CODE

Line 2 Calls routine UP.DATE to print a battle summary.

Line 3 Schedules another update in 24 hours.

X. "EVENT STOP.SIMULATION"

Event STOP.SIMULATION is called from the main program at the scheduled time to stop the simulation. It causes a final printout of the battle summary to be produced.

PERMANENT ATTRIBUTES (REAL)

TIME.V

ROUTINES CALLED

UP.DATE

PROGRAM LISTING

```
1  EVENT STOP.SIMULATION
2  LIST TIME.V
3  CALL UP.DATE
4  STOP
5  END
```

EXPLANATION OF CODE

Line 2 Prints the time the simulation ended.

Line 3 Calls routine UP.DATE to print a final battle summary.

Y. "EVENT UP.S4.AMMO"

This event is scheduled by event COM.AMMO when a resupply request is created. The purpose of event UP.S4.AMMO is to process requests for and issue ammo from

the battalion's reserve stocks of ammunition. In execution, the event performs the following functions: prioritization of outstanding requests; assessment of quantities to be released to fill requests subject to transportation availability; creation of convoys to carry the supplies; creation of cargo manifests for individual trucks; and the dispatch of convoys from the supply point. Lastly, the event schedules the convoy arrival (CO.RESUPPLY.ARR).

ARGUMENTS

ISSUEE. Argument for the routine containing the unit requesting resupply.

ISSUER. Argument for the routine containing the pointer of the supply officer updating.

GLOBAL_VARIABLES_(INTEGER)

SCODE(SUPPLY CODE) (2-d). HOLDS POINTER FOR SCL.V.ITEM.

GLOBAL_VARIABLES_(REAL)

CL1.PCT(CRITICAL LON1 PERCENT) .

CL2.PCT(CRITICAL LON2 PERCENT) .

MAX.TRIP. Maximum travel time to a unit.

MIN.TRIP. Minimum travel time to a unit.

SCODE(SUPPLY CODE) .

PERMANENT_ATTRIBUTES_(INTEGER)

N.SCONVOY(NUMBER IN SUPPLY CONVOY) .

N.SWANT.LIST(NUMBER IN SUPPLY WANT LIST) .

SCO.CDR(SUPPLY COMPANY COMMANDER).

PERMANENT_ATTRIBUTE_(REAL)

TIME.V

RECURSIVE_VARIABLES_(INTEGER).

A - Holds pointer to S-4 updating.

COM. Holds pointer to company updating.

CON.ID(CONVOY ID). Holds the pointer value of the convoy
currently being filled.

IT.LIVES. Indicates if a convoy has already been created to
carry supplies to a particular unit.

I - Loop index.

K - Loop index.

N.RNDS(NUMBER OF ROUNDS). Holds the number of rounds being
released to fill a request.

RC.TEMP(ROUND/CUBE TEMPORARY). Holds the computational value
of the number of rounds that may be loaded on a truck
due to cube restrictions.

RNDS(ROUNDS). Holds the number of rounds being released to
fill a request.

RR(RESUPPLY REQUEST). Holds the pointer of the resupply
request being currently filled.

RW.TEMP(ROUND/WEIGHT TEMPORARY). Holds the computational value of the number of rounds that may be loaded on a truck due to weight restrictions.

T.COUNT(TRUCK COUNT). Holds the value of the number of trucks already loaded for a particular resupply request.

TRKS.AVAIL (TRUCKS.AVAILABLE). Holds the total number of trucks available for assignment to a resupply mission.

CU(CUBE). Holds the maximum cube that may be loaded on a resupply vehicle.

CU.AVAIL(CUBE AVAILABLE). Holds the total cube capacity that is available within the supply unit for the shipment of supplies.

CU.REQ(CUBE REQUIRED). Holds the total cube that is required in order to ship a resupply request.

LON1. Variable which indicates whether Level of Need 1 requests are currently filed with the supply officer.

LON2. Variable which indicates whether Level of Need 2 requests are currently filed with the supply officer.

MUL1(MULTIPLIER 1). Holds the multiplier for LON 1 requests which is used to reduce the fill on such requests.

MUL2(MULTIPLIER 2). Holds the multiplier for LON 2 requests which is used to reduce the fill on such requests.

MULT(MULTIPLIER) . Holds the multiplier for both LON 1 and

LON 2 requests in the main part of the computations.

PCT(PERCENTAGE) . Holds the percentage value of the amount

released to fill a request versus the total quantity

requested.

RCU(RESIDUAL CUBE) . The cube remaining in a CONVOY to be

filled.

RWT(RESIDUAL WEIGHT) . The weight remaining in a CONVOY to be

filled.

WT(WEIGHT) . Holds the maximum weight that may be loaded on a

resupply vehicle.

WT.AVAIL(WEIGHT AVAILABLE) . Holds the total weight capacity

that is available within the supply unit for the

shipment of supplies.

WT.REQ(WEIGHT REQUIRED) . Holds the total weight that is

required in order to ship a resupply request.

ROUTINES CALLED

FILE.UPDATE LOAD.THE.TRUCK

MAX.F MIN.F

PRI.RESUPPLY TRUNC.F

UNIFORM.F UP.DATE

WT.AND.CUBE

SETS

C.CGO.LIST (COMPANY CARGO LIST) . Contains a master listing of the RES.REQs that are loaded on the trucks belonging to the set ELEMENTS (of a convoy) .

CARGO. Contains a listing of the T.CGO loaded on a particular truck.

CWANT.LIST (COMPANY WANT LIST) . Contains a listing of all outstanding RES.REQs belonging to a company.

SCONVOY (SUPPLY CONVOY) . Contains a listing of all CONVOYS the supply officer currently has active.

SWANT.LIST (SUPPLY WANT LIST) . A list of all outstanding requests the supply officer has.

TEMPORARY ENTITIES

CONVOY. An entity created to move supply trucks (TANKs) . around the battlefield with supplies. It contains the set ELEMENTS which holds the pointers of the trucks assigned to the mission.

RES.REQ (RESUPPLY REQUEST) .

T.CGO (TRUCK CARGO) . An entity created to identify the cargo loaded on a supply truck (TANK) .

TEMPORARY ATTRIBUTES (ALPHA) .

RNOMEN (RESUPPLY NOMENCLATURE) . Attribute of an SCL.V.ITEM containing the name of the particular ammo type.

STATUS. Transmits information as to where a RES.REQ is in relation to the supply system. Values can be: TOS4, ATP, TOCO.

TNOMEN (TRUCK NOMENCLATURE). Contains the name of a particular ammo type.

TEMPORARY_ATTRIBUTES_(INTEGER)

C.MV.STATE (CONVOY MOVEMENT STATE). Indicates if a convoy is in transit between supply points.

"0" indicates no

"1" indicates yes.

CO.CNVY (COMPANY CONVOY). Argument of the routine CO.RESUPPLY.ARR holding the pointer of the convoy arriving.

CONTRKS (CONVOY TRUCKS). Contains the number of trucks assigned to a convoy.

CPNTR (CONVOY POINTER). Holds the pointer value for an active CONVOY.

L.ELEMENTS (LAST ELEMENT). System variable pointing at the last truck in a CONVOY's ELEMENTS set.

M.C.CGO.LIST (MEMBER CONVOY CARGO LIST). System generated attribute which indicates whether a RES.REQ is filed in a convoy.

MANIFEST. Holds the pointer of the CONVOY a RES.REQ is loaded on.

N.C.CGO.LIST(NUMBER IN COMPANY CARGO LIST). Indicates the total number of RES.REQS filed in a CONVOY.

N.CARGO(NUMBER IN CARGO). Indicates the total number of T.CGO items loaded on a particular truck.

N.T.ALLOC(NUMBER OF TRUCKS ALLOCATED). Contains the number of trucks to be allocated to move the rounds released for a resupply request.

ONHAND. Holds the on-hand balance for rounds of a particular ammo type at the resupply point.

RAC(RESUPPLY AMMUNITION CODE).

RDS.PKG(ROUNDS PER PACKAGE).

REQUESTOR. Contains the pointer to the unit requesting resupply.

RFILL(REQUEST FILL). Holds the number of rounds released to fill a request.

RP(RELEASE POINT). Convoy termination point.

RQTY(REQUIRED QUANTITY).

RRPNTR(RESUPPLY REQUEST POINTER).

SCREEN. Indicates whether a request has been reviewed during a particular S-4 update cycle.

SP(START POINT). Convoy start point.

SPACE. Indicates if empty space remains on trucks within a CONVOY.

SPRIORITY(SUPPLY PRIORITY). Indicates the urgency of need on a RES.REQ.

TCU(TRUCK CUBE). The cube that a truck has available to be filled.

TPNTR(TRUCK POINTER).

TQTY(TRUCK QUANTITY). Holds the number of rounds within a T.CGO that are loaded on a truck.

TRAC(TRUCK AMMUNITION CODE). Of a T.CGO item.

TWT(TRUCK WEIGHT). The weight that a truck has available to be filled.

TEMPORARY_ATTRIBUTES_(REAL)

CU.PKG(CUBE PACKAGE). Of a standard package of ammo.

WT.PKG(WEIGHT PACKAGE). Of a standard package of ammo.

PROGRAM_LISTING

```
1  EVENT UP.S4.AMMO(A,COM)
2  DEFINE R.RNDS,A,I,K,COM,RW.TEMP,RC.TEMP,RNDS,
3  IT.LIVES,T.COUNT,N.RNDS,CON.ID,RR
4  AND TRKS.AVAIL AS INTEGER VARIABLES
5  DEFINE MUL1,MUL2,LON1,LON2,CU,CU.AVAIL,CU.REQ,PCT,
6  MULT,WT,WT.AVAIL,WT.REQ,RWT,RCU AS REAL VARIABLES
7  PRINT 1 LINE WITH TIME.V AS FOLLOWS
8  EVENT UP.S4.AMMO CALLED AT TIME.V = **.*
9  ''
10 CALL FILE.UPDATE(A,COM)
11 CALL WT.AND.CUBE GIVEN A
12 YIELDING WT.AVAIL,CU.AVAIL,TRKS.AVAIL,WT,CU
13 ''TEST IF RESUPPLY MISSIONS ARE POSSIBLE
14 ''
15 IF TRKS.AVAIL = 0
16     RETURN
17 OTHERWISE
18 ''
19 CALL PRI.RESUPPLY GIVEN WT.AVAIL,CU.AVAIL,A
    YIELDING MUL1,MUL2,LON1,LON2
```



```

20  ''
21  'REQUEST'
22  'CHECK RES.REQ BY LON, BY CRITICALITY, BY TIME
23  FOR I = 1 TO 5, DO
24  ''
25  'ID AND LOOP OVER REQ FROM THE MOST CRITICAL UNIT
26  FOR EVERY RES.REQ IN SWANT.LIST(A)
27  WITH SPRIORITY(RES.REQ)=I
28  AND M.C.CGO.LIST(RES.REQ) = 0, DO
29  ''
30  'CHECK SCREEN
31  IF SCREEN(RES.REQ) = 1,
32  CYCLE
33  OTHERWISE
34  LET SCREEN(RES.REQ) = 1
35  ''
36  'DETER IF THE AMMO IS ON-HAND AT SUPPLY POINT
37  IF ONHAND(SCODE(A,RAC(RES.REQ))) EQ 0
38  CYCLE
39  ALWAYS
40  ''
41  'SET MULTIPLIERS
42  LET MULT = 1.0
43  IF SPRIORITY(RES.REQ) = 1,
44  LET MULT = MUL1
45  ALWAYS
46  IF SPRIORITY(RES.REQ) = 2,
47  LET MULT = MUL2
48  ALWAYS
49  ''
50  'DETER IF THERE IS ENOUGH AMMO TO MEET REQMENTS
51  LET N.RNDS = RQTY(RES.REQ) * MULT
52  LET R.RNDS = RQTY(RES.REQ)
53  IF N.RNDS GT ONHAND(SCODE(A,RAC(RES.REQ)))
54  LET N.RNDS = ONHAND(SCODE(A,RAC(RES.REQ)))
55  ALWAYS
56  ''
57  'DETERMINE THE WT AND CUBE TO BE SHIPPED
58  LET WT.REQ=N.RNDS*WT.PKG(SCODE(A,RAC(RES.REQ)))
59  LET CU.REQ= N.RNDS*CU.PKG(SCODE(A,RAC(RES.REQ)))
60  LET CU.REQ= N.RNDS*CU.PKG(SCODE(A,RAC(RES.REQ)))
61  LET CU.REQ= N.RNDS*CU.PKG(SCODE(A,RAC(RES.REQ)))
62  ''
63  'DETER IF A UNIT CONVOY IS ALREADY FORMED
64  FOR EVERY CONVOY IN SCONVOY(A)
65  WITH C.MV.STATE(CONVOY) = 0 AND SP(CONVOY) = A
66  AND RP(CONVOY) = REQUESTOR(RES.REQ), DO
67  'CAPTURE THE POINTER VALUE OF THE CONVOY
68  LET CON.ID = CONVOY
69  LET IT.LIVES = 1
70  ''
71  'CHECK IF SPACE IS AVAIL ON CONVOY TRUCKS
72  IF SPACE(CONVOY) = 1
73  IF WT.REQ GT TWT(L.ELEMENTS(CONVOY))
74  OR CU.REQ GT TCU(L.ELEMENTS(CONVOY))
75  LET RWT = TWT(L.ELEMENTS(CONVOY))
76  LET RCU = TCU(L.ELEMENTS(CONVOY))
77  LET RW.TEMP=RWT
78  LET RW.TEMP=RW.TEMP / WT.PKG(SCODE(A,RAC(RES.REQ)))
79  LET RW.TEMP=RW.TEMP * RDS.PKG(SCODE(A,RAC(RES.REQ)))
80  LET RC.TEMP = RCU
81  LET RC.TEMP=RCU / CU.PKG(SCODE(A,RAC(RES.REQ)))
82  LET RC.TEMP=RC.TEMP * RDS.PKG(SCODE(A,RAC(RES.REQ)))
83  LET RNDS = MIN.F(RW.TEMP,RC.TEMP)
84  LET SPACE(CONVOY) = 0
85  ELSE
86  LET RNDS = N.RNDS
87  ALWAYS

```



```

88      ''
89      IF RNDS le 0,
90      go to endfill
91      otherwise
92      CREATE A T.CGO
93      LET TPNTR(T.CGO) = L.ELEMENTS(CONVOY)
94      LET RRPNTR(T.CGO) = RES.REQ
95      LET TNOMEN(T.CGO) = RNOMEN(RES.REQ)
96      LET TRAC(T.CGO) = RAC(RES.REQ)
97      LET TOTY(T.CGO) = RNDS
98      FILE T.CGO IN CARGO(L.ELEMENTS(CONVOY))
99      ''REDUCE THE QUANTITY ON THE RES.REQ
100     LET N.RNDS = N.RNDS - RNDS
101     LET RFILL(RES.REQ) = RFILL(RES.REQ) + RNDS
102     LET RQTY(RES.REQ) = RQTY(RES.REQ) - RNDS
103     ''REDUCE THE ON-HAND BALANCE OF STOCKS
104     LET ONHAND(SCODE(A,RAC(RES.REQ))) =
105     ONHAND(SCODE(A,RAC(RES.REQ))) - RNDS
106     ''REDUCE THE WEIGHT AND CUBE FOR THE REQ
107     LET WT.REQ= TRUNC.F(WT.REQ - RNDS
108     *WT.PKG(SCODE(A,RAC(RES.REQ))))
109     /RDS.PKG(SCODE(A,RAC(RES.REQ))))
110     LET CU.REQ= TRUNC.F(CU.REQ - RNDS
111     *CU.PKG(SCODE(A,RAC(RES.REQ))))
112     /RDS.PKG(SCODE(A,RAC(RES.REQ))))
113     ''REDUCE THE WT AND CUBE AVAIL ON THE
114     TRUCK
115     LET TWT(L.ELEMENTS(CONVOY)) =
116     TRUNC.F(TWT(L.ELEMENTS(CONVOY))
117     - RNDS*WT.PKG(SCODE(A,RAC(RES.REQ))))
118     /RDS.PKG(SCODE(A,RAC(RES.REQ))))
119     LET TCU(L.ELEMENTS(CONVOY)) =
120     TRUNC.F(TCU(L.ELEMENTS(CONVOY))
121     - RNDS*WT.PKG(SCODE(A,RAC(RES.REQ))))
122     /RDS.PKG(SCODE(A,RAC(RES.REQ))))
123     ALWAYS
124     ''
125     ''FILL IN CONVOY MANIFEST
126     IF CON.ID IS NOT IN SOME SCONVOY
127     FILE CON.ID IN SCONVOY(A)
128     ALWAYS
129     IF RES.REQ NOT IN C.CGO.LIST,
130     FILE RES.REQ IN C.CGO.LIST(CON.ID)
131     ALWAYS
132     LET MANIFEST(RES.REQ) = CON.ID
133     LET STATUS(RES.REQ) = "TOCOMPANY"
134     IF RQTY(RES.REQ) = 0
135     GO TO REQUESTLOOP
136     OTHERWISE
137     ALWAYS
138     'END.SPACE.CHECK' LOOP
139     ''
140     'RALLY'
141     ''CHECK NUMBER OF TRUCKS AVAIL FOR SHIPMENT
142     IF TRKS.AVAIL EQ 0
143     GO TO FINALCHECK
144     OTHERWISE
145     ''
146     ''IF A CONVOY DOESN'T EXIST CREATE ONE
147     IF IT.LIVES EQ 0
148     CREATE AN CONVOY
149     LET CPNTR(CONVOY) = CONVOY
150     LET CON.ID = CONVOY
151     LET RP(CONVOY) = REQUESTOR(RES.REQ)
152     LET SP(CONVOY) = A
153     ALWAYS
154     ''
155     ''DETERMINE THE # OF TRKS ARE ON-HAND

```



```

152      '' IF NOT ADJUST
153      IF WT.REQ LT WT.AVAIL AND CU.REQ LT CU.AVAIL,
154      LET N.T.ALLOC(RES.REQ) =
155      TRUNC.F(MAX.F(WT.REQ/WT, CU.REQ/CU) + 1)
156      ELSE
157      LET N.T.ALLOC(RES.REQ) = TRKS.AVAIL
158      ALWAYS
159      LET TRKS.AVAIL = TRKS.AVAIL - N.T.ALLOC(RES.REQ)
160      ''
161      '' FILL IN CONVOY MANIFEST
162      IF CON.ID IS NOT IN SOME SCONVOY
163      FILE CON.ID IN SCONVOY(A)
164      ALWAYS
165      IF RES.REQ NOT IN C.CGO.LIST,
166      LET STATUS(RES.REQ) = "TO COMPANY"
167      FILE RES.REQ IN C.CGO.LIST(CON.ID)
168      ALWAYS
169      LET MANIFEST(RES.REQ) = CON.ID
170      ADD N.T.ALLOC(RES.REQ) TO CONTRKS(CON.ID)
171      LET RR = RES.REQ
172      CALL LOAD.THE.TRUCKS
173      (A, RR, WT.REQ, CU.REQ, N.RNDS, CON.ID)
174      ''
175      '' ADJUST THE WT. AND CUBE AVAIL FOR SHIPPING
176      CALL WT.AND.CUBE GIVEN A
177      YIELDING WT.AVAIL, CU.AVAIL, TRKS.AVAIL, WT, CU
178      ''
179      'REQUESTLOOP'
180      '' UPDATE SWANT.LIST FILES
181      LET PCT = R.RNDS/ROTY(RES.REQ)
182      IF (SPRIORITY(RES.REQ) = 1 AND PCT GT C.L1PCT)
183      OR (SPRIORITY(RES.REQ) = 2 AND PCT GT C.L2PCT)
184      REMOVE RES.REQ FROM SWANT.LIST(A)
185      ELSE
186      REMOVE RES.REQ FROM SWANT.LIST(A)
187      ALWAYS
188      ''
189      LET IT.LIVES = 0
190      ''
191      LOOP
192      LOOP
193      ''
194      '' DISPATCH ALL CONVOYS CREATED
195      FOR EVERY CONVOY IN SCONVOY(A)
196      WITH C.MV.STATE(CONVOY) = 0 AND SP(CONVOY) = A, DO
197      LET C.MV.STATE(CONVOY) = 1
198      LOOP
199      ''
200      IF CON.ID NE 0
201      SCHEDULE A CO.RESUPPLY.ARR(CON.ID)
202      IN UNIFORM.F(MINTRIP, MAXTRIP, TSTREAM) MINUTES
203      ALWAYS
204      ''
205      '' RESET LOOP CHECKS
206      FOR ALL RES.REQ IN CWANT.LIST(SCO.CDR(A)), DO
207      LET SCREEN(RES.REQ) = 0
208      LOOP
209      RETURN
210      END

```

EXPLANATION OF CODE

Lines 2-6 Define recursive variables for the routine.

Line 7 Prints a message marking the beginning of the event.

Line 9 Calls routine FILE.UPDATE which files new requests
 and checks old requests for duplication.

Lines 10-11 Calls routine WEIGHT.AND.CUBE to assess the
 total weight and cube capacity available to handle
 shipments of supplies forward.

Lines 13-16 Check if all resupply vehicles are already in
 use. If so, the routine is ended with no further action
 taken.

Line 18-19 Calls routine PRI.RESUPPLY to determine if LON1
 and LON2 requests should be modified in order to fill
 more requests with a lesser amount of ammo.

Lines 21-23 Start the major loop of the routine. Initialize
 an index to the most urgent LON a resupply request can
 have and begin the loop which will fill requests. Loop
 ends on line 192.

Lines 25-28 Start an inner loop of the routine to identify
 the most critical resupply request on-hand to fill
 first. Loop ends on line 191.

Lines 30-34 Check the screen attribute of a resupply
 request to determine if the request has been reviewed
 before in the current cycle. If the request has been
 reviewed, the request is cycled.

Lines 36-39 Check the availability of the ammo requested at the supply point. If none is available the request is cycled.

Lines 41-48 Set multipliers for high priority requests as appropriate. The purpose of the multipliers is to reduce the fill on high priority requests in order to release truck space to fill more high priority requests.

Lines 50-55 Determine if there is enough ammo available to fill requests. If not, the request is filled with what is available.

Lines 57-61 Determine the weight and cube needed to ship the number of rounds requested.

Lines 63-66 Start an inner loop which checks to see if a convoy is already formed for requestor of the current request. If one exists, this loop is entered and any available space on trucks already committed to the convoy is filled before more trucks are committed. Loop ends on line 134.

Line 68 Captures the pointer value of the convoy formed.

Line 69 Initializes a variable signifying that a convoy is already formed for the unit.

Lines 72-128 Perform an IF check to determine if space is still available on convoy trucks. If so, the logic embedded to line 133 is executed; if not, control is transferred to line 133.

Lines 73-87 Perform an IF check to compute the number of rounds that may be shipped on the convoy subject to the available weight and cube. Satisfying the IF condition indicates that the current request will fill the room available. Transfer to the ELSE condition indicates that the request will leave space on trucks for additional requests.

Line 75 Computes the residual weight available on the convoy.

Line 76 Computes the residual cube available on the convoy.

Lines 77-79 Compute the rounds that may be shipped for a request subject to the weight limitations of the truck.

Lines 80-82 Compute the rounds that may be shipped for the request subject to the cube limitations of the truck.

Line 83 Specifies the number of rounds to be released subject to the minimum restriction.

Line 84 Sets the space attribute of the convoy to indicate that there is no space available.

Lines 85-86 Mark the ELSE logic which indicates that the request may be filled with room to spare.

Line 87 Marks the end of the weight and cube loop.

Lines 89-121 Perform an IF check to see if rounds are to be loaded on committed convoy trucks. If RNDS is greater than zero, the logic to load the truck is executed. If not, control is passed to line 133.

Line 92 Creates a T.CGO to hold information as to the type and quantity of ammo to be loaded on a truck.

Line 93 Captures the pointer value of the truck the cargo is loaded on.

Line 94 Captures the pointer value of the resupply request being filled.

Line 95 Captures the nomenclature of the request being filled.

Line 96 Captures the ammunition code of the request being filled.

Line 97 Captures the quantity being loaded on the truck.

Line 98 Files the cargo on the last truck in the convoy.

Lines 99-100 Reduce the number of rounds yet to be filled for the request.

Line 101 Adds the quantity released to the fill attribute of the request.

Line 102 Reduces the required quantity for the resupply request.

Lines 103-105 Reduce the on-hand stocks for the ammo type at the supply point.

Lines 106-111 Reduce the weight and cube required to fill the request.

Lines 113-120 Reduce the weight and cube available on the truck to fill requests.

Line 121 Closes out the inner IF check.

Lines 121-132 Fill in the convoy manifest.

Lines 125-127 Check if the existing convoy is in the set of convoys owned by the supply officer. If not, the convoy is filed.

Line 128 Points the resupply request to the convoy it is loaded on.

Line 129 Insures that the status of the convoy shows it enroute to the supported unit.

Lines 130-132 Check to see if the request has been filled. If so the loop is cycled to the next request. If not, logic continues to see if it can be loaded on an additional truck for the convoy.

Lines 136-140 Determine if there are trucks available for assignment to the convoy. If not, all loops are terminated.

Lines 142-149 Make a check to determine if a convoy is already formed to ship supplies to the unit. If not a convoy is formed, and attributes are set capturing its pointer value, start point, release point(end point), and setting a variable equal to its pointer value for later computations.

Lines 151-159 Determine if the necessary number of trucks are available to ship the supplies. If not, the number of trucks that are available are assigned to the mission.

Lines 161-173 Fill in the convoy manifest.

Lines 162-164 Check if the existing convoy is in the set of convoys owned by the supply officer. If not, the convoy is filed.

Lines 165-168 Check if the resupply request is filed in the convoy manifest. If not, the convoy is filed and the status changed to show enroute to the supported unit.

Line 169 Points the resupply request to the convoy it is loaded on.

Line 170 Adds the number of trucks allocated to the
convoy's total.

Lines 171-172 Capture the resupply pointer and transfer
control to a routine which loads the request on the
trucks.

Lines 175-177 Call routine WEIGHT.AND.CUBE again to update
the weight and cube now available on trucks at the
battalion trains for shipping.

Lines 179-187 Update S-4 request files dropping those LON 1
and LON 2 files that have been filed past a critical
minimum and dropping all other requests that have at
least a partial fill.

Line 189 Resets the IT.LIVES variable to zero.

Line 191 Closes out the inner loop started on line 25
carrying with it the most critical request. Transfers
control back to acquire the next request or out to the
next LON value.

Line 192 Closes out the major loop started on line 21.
Transfers control back to change the LON value being
considered to the next value or out.

Lines 194-198 Change the movement state of all convoys
created in order to dispatch the convoys.

Lines 200-203 Determine if convoys have been formed this iteration of the routine. If so, a company resupply arrival time is scheduled.

Lines 205-208 Reset the screen attribute on still viable requests for the next iteration of the loop.

Z. "EVENT FIREKILL"

This event is scheduled by event UP.W.AMMO if a weapon system sustains a firepower kill. The purpose of the event is to redistribute the weapon system's on-board ammo to the other members of its platoon. In execution, each of the six ammo types a weapon could carry are removed from the weapon and distributed to the other undamaged weapons in the platoon in accordance with each weapon's urgency of need for that ammo type.

ARGUMENT

VICTIM. Points to the weapon system that has been killed.

DEFINE TO MEAN

AMMO1. AP.TOW(ARMOR PIERCING/TOW ROUNDS).

AMMO2. HE.DRAG(HEAT/DRAGON ROUNDS).

AMMO3. AW1.OR.MSL3(ALTERNATE WEAPON 1 OR MISSILE 3).

AMMO4. AW2.OR.ADM(ALTERNATE WEAPON 2 OR AIR DEFENSE MISSILE).

GLOBAL VARIABLES (INTEGER)

PCODE (PLATOON CODE) (2-d). Holds the pointer value for a platoon's PCL.V.ITEMS (PLATOON CLASS V ITEMS).

RECURSIVE VARIABLES (INTEGER)

A - Points to the weapon killed.

AC. Holds the value of the ammo code being reviewed.

DEL. Placeholder for ammunition types being delivered.

I - Loop index.

NO.BATTLE. Indicates whether RES.REQs should be filled.

"0" indicates no

"1" indicates yes

PL. Points to a PLATOON.LEADER.

TK. Points to a TANK.

ROUTINES CALLED

UP.DATE, W.AMMO, P.CLASS.V

SETS USED

PLAT.UNIT (PLATOON UNIT). Owned by a platoon.leader. Members are the unit's combat vehicles (TANKs).

TEMPORARY ATTRIBUTES (INTEGER)

AMMO5 (AMMUNITION 5). Of TANK.

AMMO6 (AMMUNITION 6). Of TANK.

AP.TOW (ARMOR-PIERCING/TOW). Ammunition 1 of TANK.

AW1.OR.MSL3 (ALTERNATE WEAPON 1 OR MISSILE 3). Ammunition 3.

AW2.OR.ADM (ALTERNATE WEAPON 2 OR AIR DEFENSE MISSILE) .

Ammunition 4.

HE.DRAG (HEAT/DRAGON ROUNDS) . Ammunition 2 of TANK.

FKILL (FIREPOWER KILL) .

FLAG. Yielding argument of routine W.AMMO; not used.

KKILL (CATASTROPHIC KILL) .

MPKILL (MOBILITY AND FIREPOWER KILL) .

MKILL (MOBILITY KILL) .

OH1 (ON-HAND 1) . Current balance of ammunition 1 on a TANK.

OH2 (ON-HAND 2) . Current balance of ammunition 2 on a TANK.

OH3 (ON-HAND 3) . Current balance of ammunition 3 on a TANK.

OH4 (ON-HAND 4) . Current balance of ammunition 4 on a TANK.

OH5 (ON-HAND 5) . Current balance of ammunition 5 on a TANK.

OH6 (ON-HAND 6) . Current balance of ammunition 6 on a TANK.

P.SHORT (PLATOON SHORTAGE) . Current shortage of a PCL.V.ITEM

(PLATOON CLASS V ITEM) . Unique to each platoon and ammo
type.

PLTLDR (PLATOON LEADER) .

SLOAD1 (STOWED LOAD 1) . Optimal load ammo type 1.

SLOAD2 (STOWED LOAD 2) . Optimal load ammo type 2.

SLOAD3 (STOWED LOAD 3) . Optimal load ammo type 3.

SLOAD4 (STOWED LOAD 4) . Optimal load ammo type 4.

SLOAD5 (STOWED LOAD 5) . Optimal load ammo type 5.

SLOAD6 (STOWED LOAD 6) . Optimal load ammo type 6.

SYS.TYPE (SYSTEM TYPE) .

TAC1. (TANK AMMUNITION CODE 1) .

TAC2. (TANK AMMUNITION CODE 2) .

TAC3. (TANK AMMUNITION CODE 3) .

TAC4. (TANK AMMUNITION CODE 4) .

TAC5. (TANK AMMUNITION CODE 5) .

TAC6. (TANK AMMUNITION CODE 6) .

TEMPORARY ENTITIES

TANK

PROGRAM LISTING

```
1  EVENT FIREKILL(A)
2  DEFINE NO.BATTLE,FLAG,TK,PL,DEL,AC,A,I
3  AS INTEGER VARIABLES
4  ''
5  PRINT 1 LINE THUS
   EVENT FIREPOWER KILL CALLED
6  ''
7  ''CHANGE KILL STATUS TO ELIMINATE THE WEAPON
8  FROM FURTHER UPDATES
9  LET MKILL(A) = 1
10 ''
11 ''
12 FOR I = 1 TO 6, DO
13   ''SET UP ARTIFICIAL DELIVERY
14   ''
15   GO TO 1,2,3,4,5,6 PER I
16   '1'
17   LET DEL = AMMO1(A)
18   LET AC = TAC1(A)
19   '2'
20   LET DEL = AMMO2(A)
21   LET AC = TAC2(A)
22   '3'
23   LET DEL = AMMO3(A)
24   LET AC = TAC3(A)
25   '4'
26   LET DEL = AMMO4(A)
27   LET AC = TAC4(A)
28   '5'
29   LET DEL = AMMO5(A)
30   LET AC = TAC5(A)
31   '6'
32   LET DEL = AMMO6(A)
33   LET AC = TAC6(A)
34   ''
```



```

35  ''FILL AS
36  FOR EVERY A IN PLAT.UNIT(PLTLDR), DO
37    IF MKILL(A)=1 OR MFKILL(A)=1 OR FKILL(A)=1
38    OR KKILL(A)=1
39    CYCLE
40    OTHERWISE
41  ''
42    IF TAC1(A)=AC
43      LET OH1(A)=OH1(A)+DEL( (SLOAD1(A)-OH1(A)) ) /
44        P.SHORT(PCODE(PLTLDR,AC))
45      LET AMMO1(A)=AMMO1(A)+DEL
46        ( (SLOAD1(A)-AMMO1(A)) ) /
47        P.SHORT(PCODE(PLTLDR,AC))
48      GO TO TKLOOP
49    OTHERWISE
50  ''
51    IF TAC2(A)=AC
52      LET OH2(A)=OH2(A)+DEL( (SLOAD2(A)-OH2(A)) ) /
53        P.SHORT(PCODE(PLTLDR,AC))
54      LET AMMO2(A)=AMMO2(A)+DEL
55        ( (SLOAD2(A)-AMMO2(A)) ) /
56        P.SHORT(PCODE(PLTLDR,AC))
57      GO TO TKLOOP
58    OTHERWISE
59  ''
60    IF TAC3(A)=AC
61      LET OH3(A)=OH3(A)+DEL( (SLOAD3(A)-OH3(A)) ) /
62        P.SHORT(PCODE(PLTLDR,AC))
63      LET AMMO3(A)=AMMO3(A)+DEL
64        ( (SLOAD3(A)-AMMO3(A)) ) /
65        P.SHORT(PCODE(PLTLDR,AC))
66      GO TO TKLOOP
67    OTHERWISE
68  ''
69    IF TAC4(A)=AC
70      LET OH4(A)=OH4(A)+DEL( (SLOAD4(A)-OH4(A)) ) /
71        P.SHORT(PCODE(PLTLDR,AC))
72      LET AMMO4(A)=AMMO4(A)+DEL
73        ( (SLOAD4(A)-AMMO4(A)) ) /
74        P.SHORT(PCODE(PLTLDR,AC))
75      GO TO TKLOOP
76    OTHERWISE
77  ''
78    IF TAC5(A)=AC
79      LET OH5(A)=OH5(A)+DEL( (SLOAD5(A)-OH5(A)) ) /
80        P.SHORT(PCODE(PLTLDR,AC))
81      LET AMMO5(A)=AMMO5(A)+DEL
82        ( (SLOAD5(A)-AMMO5(A)) ) /
83        P.SHORT(PCODE(PLTLDR,AC))
84      GO TO TKLOOP
85    OTHERWISE
86  ''
87    IF TAC6(A)=AC
88      LET OH6(A)=OH6(A)+DEL( (SLOAD6(A)-OH6(A)) ) /
89        P.SHORT(PCODE(PLTLDR,AC))
90      LET AMMO6(A) = AMMO6(A)+DEL
91        ( (SLOAD6(A)-AMMO6(A)) ) /
92        P.SHORT(PCODE(PLTLDR,AC))
93  ALWAYS
94  'TKLOOP' LOOP
95  LOOP
96  FOR EVERY TK IN PLAT.UNIT(PLTLDR(A)), DO
97    IF SYS.TYPE(TK) EQ 7,
98    CYCLE
99    OTHERWISE
100    LET NO.BATTLE = 1
101    CALL W.AMMO(TK,NO.BATTLE) YIELDING FLAG
102  LOOP

```



```
103  ''  
104  CALL P.CLASS.V(PLTLDR(A))  
105  RETURN  
106  END
```

EXPLANATION_OF_CODE

Line 2-3 Define recursive variables used in the routine.

Line 5 Prints a message indicating that the routine has been called.

Lines 7-9 Change the kill status of the weapon system to eliminate it from further supply computations.

Line 12 Begins the major loop for the routine, looping over the six ammo types carried on a weapon and distributing these assets to the other members of the platoon. Loop ends on line 95.

Lines 16-33 Set DEL equal to the amount being taken from the damaged weapon and AC equal to its identifying ammo code.

Lines 36 Begins an inner loop over the undamaged weapons of the platoon in order to distribute the damaged vehicles ammo. Loop ends on line 95.

Lines 37-40 Check the weapon being considered for any damage and cycles if damage is determined.

Lines 43-93 Loop over the six ammo types carried by the undamaged weapon to see if it carries the ammo being

distributed from the damaged vehicle. If a match is made, the amount delivered is equal to the amount being taken from the damaged vehicle times the ratio of the weapon's need to the platoon's overall need.

Line 94 Closes the weapon system loop begun on line 36.

Control is transferred either to the next weapon or to the next ammo.

Line 95 Closes the ammo loop begun on line 12. Control is transferred either to the next ammo type or out.

Lines 96-102 Cause every weapon system to update its ammo status.

Line 104 Causes the platoon to update its ammo status.

LIST OF REFERENCES

1. Armament Systems Inc., Human Engineering Laboratory Ammunition Point Simulation (HELAPS-2), U.S. Army Missile and Munitions Center and School, Director of Combat Developments, Redstone Arsenal, Alabama. August, 1981.
2. Remen D.J., Clarke R.B., Fox J., Ammunition Resupply Model (ARM), Technical Report TR 2-80 United States Army Combined Arms Center, Fort Leavenworth, Kansas, March 1980.
3. Kelley, John R., Simulation and Analysis of Ammunition Transport Capability in Support of a Combat Unit, Master's Thesis, Naval Postgraduate School, Monterey, California, 1978.
4. Wallace, William S. and Hagewood, Eugene G., Simulation of Tactical Alternative Responses (STAR), Master's Thesis, Naval Postgraduate School, Monterey, California, June, 1978.
5. Ripley, Bruce G., A Dynamic Ammunition and Fuel Resupply Model in Support of the STAR Model, Master's Thesis, Naval Postgraduate School, Monterey, California, 1979.
6. Kirby D.G. and Schultz D.P., A High Resolution Integrated Combat and Logistics Model (STAR-LOG), Master's Thesis, Naval Postgraduate School, Monterey, California, March 1980.
7. Field Manual No. 9-6, Ammunition Service in the Theatre of Operations, Headquarters Department of the Army, Washington, DC., 31 July 1981.

BIBLIOGRAPHY

Carpenter, H.J. and Thurman E.E., Parametric Simulation of Infantry Tactics and Equipment (Dismounted STAR) Master's Thesis, Naval Postgraduate School, Monterey, California, June 1978.

Hartman, J.K., Ground Movement Modelling in the STAR Combat Model Naval Postgraduate School, Technical Report NPS 55-80-021, Monterey, California, May 1980.

Kiviat, P.J., Villanueva, R., Markowitz, H.M., SIMSCRIPT II.5 Programming Language 3d Edition, Consolidated Analysis Center, Inc., 1975.

Loerch, A.G., Parametric Simulation of the Direct Support Maintenance System in a Brigade Area Master's Thesis, Naval Postgraduate School, Monterey, California, September 1980.

Parry, S.H. and Kelleher, E.P., Tactical Parameters and Input Requirements for the Ground Component of the STAR Combat Model Naval Postgraduate School, Technical Report NPS 5555-79-023, Monterey, California, October 1979.

Taylor, J.G., Force-on-Force Attrition Modelling Naval Postgraduate School, Monterey, California, January 1978.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3. Department Chairman, Code 55 Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
4. Chief TRADOC Research Element Monterey Naval Postgraduate School Monterey, California 93940	1
5. Associate Professor James K. Hartman Code 55Hh Department of Operations Research Naval Postgraduate School Monterey, California 93940	5
6. Associate Professor S. H. Parry Code 55Py Department of Operations Research Naval Postgraduate School Monterey, California 93940	5
7. Headquarters US Army Training and Doctrine Command ATTN: Director, Studies and Analysis Directorate, Mr. S. Goldberg Fort Monroe, Virginia 23651	1
8. Associate Professor A. L. Schoenstadt Code 53Zh Department of Mathematics Naval Postgraduate School Monterey, California 93940	1
9. Professor James G. Taylor Code 55Tw Department of Operations Research Naval Postgraduate School Monterey, California 93940	1

10. Office of the Commanding General 1
US Army Training and Doctrine Command
ATTN: General Glenn Otis
Fort Monroe, Virginia 23651
11. Headquarters 1
US Army Training and Doctrine Command
ATTN: ATCG-T (BG Morelli)
Fort Monroe, Virginia 23651
12. Office of the Commanding General 1
US Readiness Command
ATTN: General Donn A Starry
MacDill AFB, Florida 33621
13. Mr. Walter Hollis 1
Deputy Under Secretary of the Army
(Operations Research)
Department of the Army, The Pentagon
Washington, D. C. 20310
14. LTG Howard Stone 1
Commanding General
US Army Combined Arms Center
Fort Leavenworth, Kansas 66027
15. Director 1
Combined Arms Combat Development Activity
ATTN: ATZL-CAC-A (Mr. Lee Pleger)
Fort Leavenworth, Kansas 66027
16. Director 1
Combat Analysis Office
ATTN: Mr. Kent Pickett
US Army Combined Arms Center
Fort Leavenworth, Kansas 66027
17. Command and General Staff College 1
ATTN: Education Advisor
Room 123, Bell Hall
Fort Leavenworth, Kansas 66027
18. Dr. Wilbur Payne, Director 1
US Army TRADOC Systems Analysis Activity
White Sands Missile Range, New Mexico 88002
19. Headquarters, Department of the Army 1
Office of the Deputy Chief of Staff for
Operations and Plans
ATTN: DAMO-2D
Washington, D.C. 20310
20. Commander 1
US Army Concepts and Analysis Agency
ATTN: MOCA-WG
8120 Woodmont Avenue
Bethesda, Maryland 20014

21. Director 1
US Army Material Systems Analysis Activity
ATTN: DRKSY-CM (Mr. Bill Niemeyer)
Aberdeen Proving Grounds, Maryland 21005
22. Director 1
US Army Night Vision and Electro-Optical
Lab
ATTN: DEL-NV-VI (Mr. Frank Shields)
Fort Belvoir, Virginia 22060
23. Director 1
USATRASANA
ATTN: Mr. Ray Heath
White Sands Missile Range, New Mexico 88002
24. Director 1
Combat Developments, Studies Division
ATTN: MAJ W. Scott Wallace
US Army Armor Agency
Fort Knox, Kentucky 40121
25. Commandant 1
US Army Field Artillery School
ATTN: ATSA-CD-DSWS
Fort Sill, Oklahoma 73503
26. Director 1
Combat Developments
US Army Aviation Agency
Fort Rucker, Alabama 36362
27. Director 1
Combat Developments
US Army Infantry School
Fort Benning, Georgia 31905
28. Director 1
Combat Developments
ATTN: ATZA-CDE (CPT James Mudd)
US Army Engineer School
Fort Belvoir, Virginia 22060
29. Director 1
Combat Developments
ATTN: ATSA-CDF-S
US Army Air Defense Agency
Fort Bliss, Texas 79905
30. Commander 1
US Army Logistics Center
ATTN: ATCL-OS (Mr. Cammeron/CPT McGrann)
Fort Lee, Virginia 23801

- | | | |
|-----|---|---|
| 31. | Commandant
US Army Signal School
ATTN: LTC Harnagel
Fort Gordon, Georgia 30905 | 1 |
| 32. | Deputy Under Secretary of the Army
for Operations Research
Room 2E261, Pentagon
Washington, DC 20310 | 1 |
| 33. | CPT Peter J. Bucha, USA
801 South Drive
Baldwin, New York 11510 | 5 |
| 34. | CPT Thomas J. McGrann, USA
8011-10th Ave.
Brooklyn, New York 11228 | 5 |

Thesis
B82945 Bucha
c.1

107071

A high resolution
ammunition resupply
model.

Thesis
B8294
c.1

25 SEP 83

38195

Thesis
B82945 Bucha
c.1

107071

A high resolution
ammunition resupply
model.

thesB82945
A high resolution ammunition resupply mo



3 2768 002 07869 3
DUDLEY KNOX LIBRARY